Policy-based signatures

Georg Fuchsbauer
IST Austria

joint work with Mihir Bellare

available on eprint 2013/413

University of Bristol, 29 Oct 2013

Overview

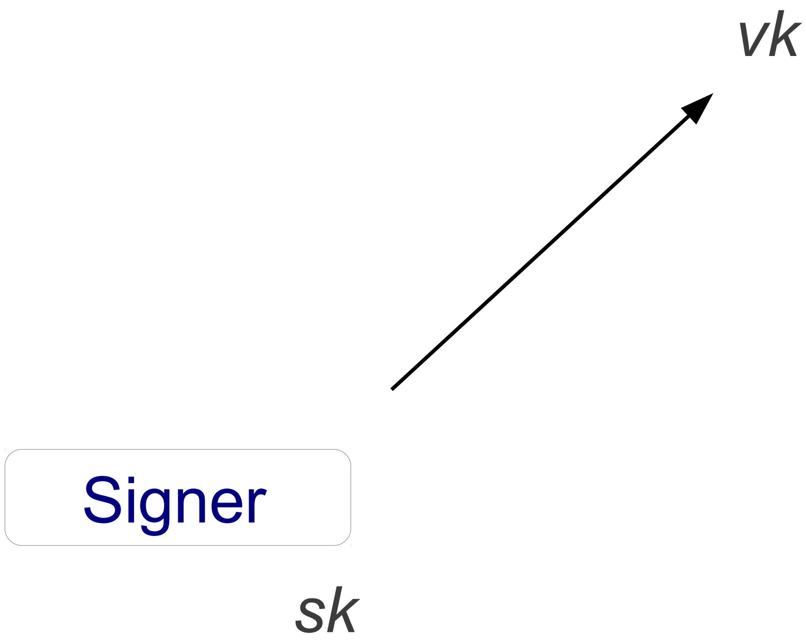
- New signature primitive
- Signer can only sign messages conforming to policy

Overview

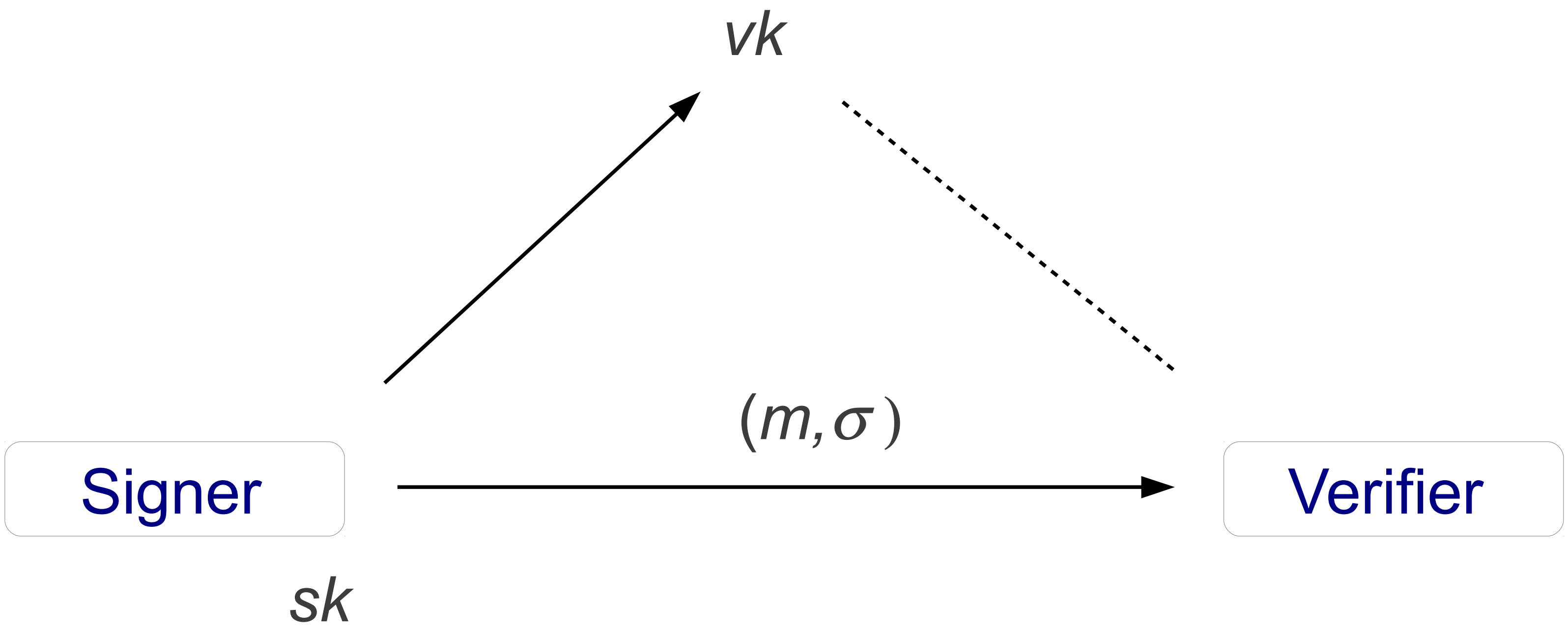
- New signature primitive
- Signer can only sign messages conforming to policy

- Practical applications: use for corporations
- Theoretical: unification of existing work

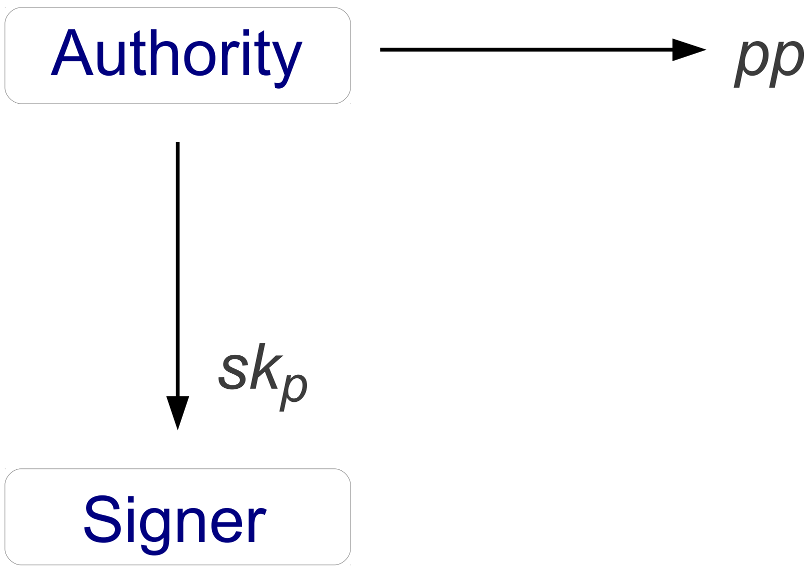
Signatures



Signatures

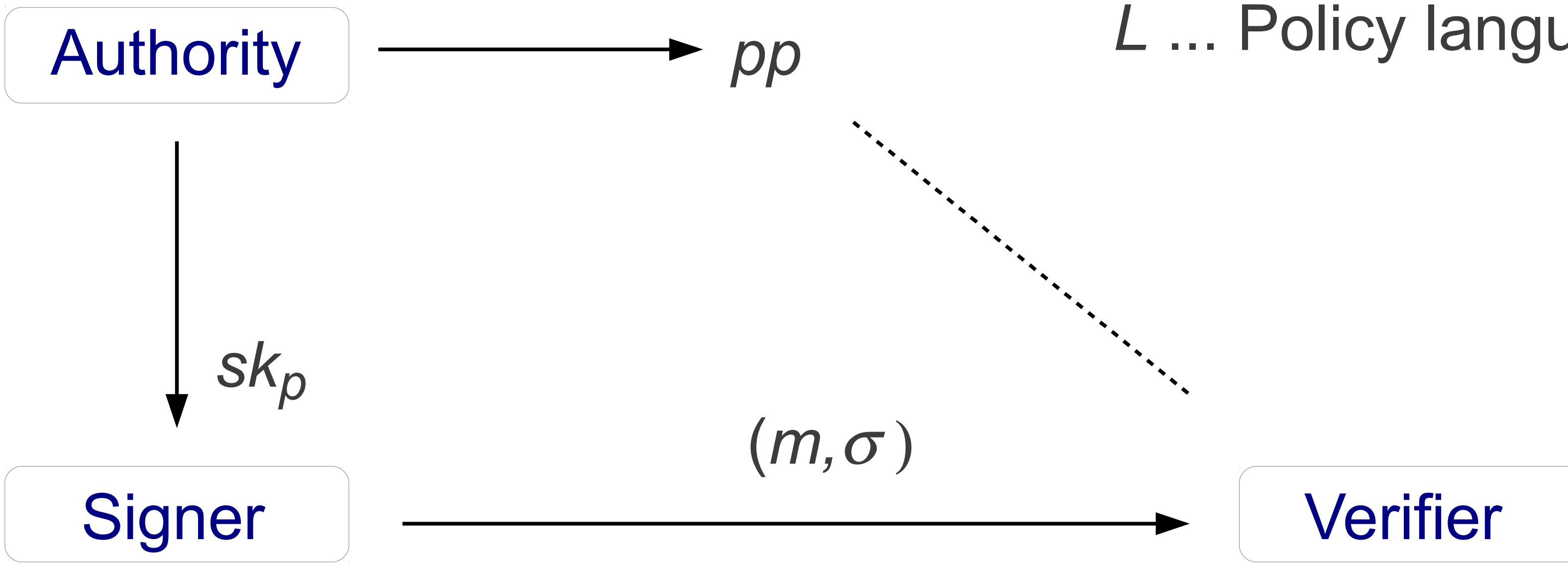


Policy-based signatures



Policy-based signatures

p ... Policy
L ... Policy language



only if $(p,m) \in L \subseteq \{0,1\}^* \times \{0,1\}^*$

Security

- Unforgeability:

  You can only sign a message *m* if you have a key for a policy *p* satisfied by *m*

Security

- Unforgeability:

  You can only sign a message *m* if you have a key for a policy *p* satisfied by *m*

- Privacy:

  The signature does not reveal the policy

Theoretical Motivation

- Signature analog to functional encryption [BSW11]

Theoretical Motivation

- Signature analog to functional encryption [BSW11]
- Unification of existing notions for signatures with privacy:

  Group signatures [Cv91]

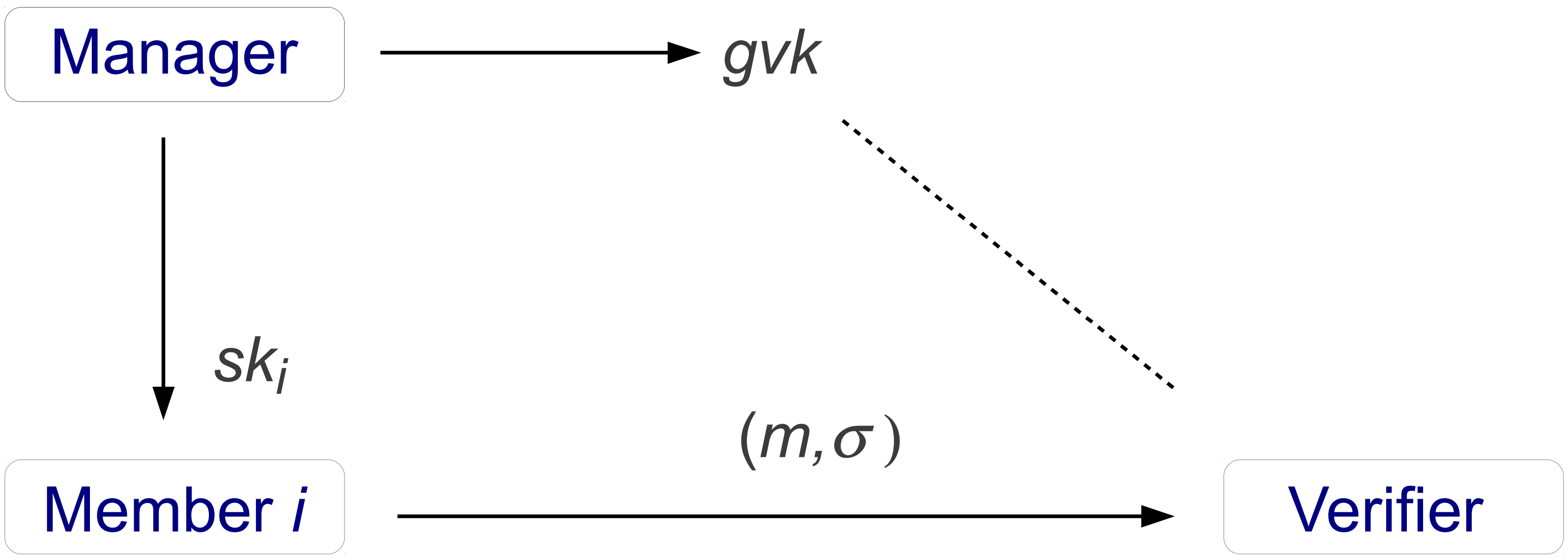
  (Anonymous) proxy signatures [MUO96,FP08]

  Ring signatures, mesh signatures [RST01,Boy07]

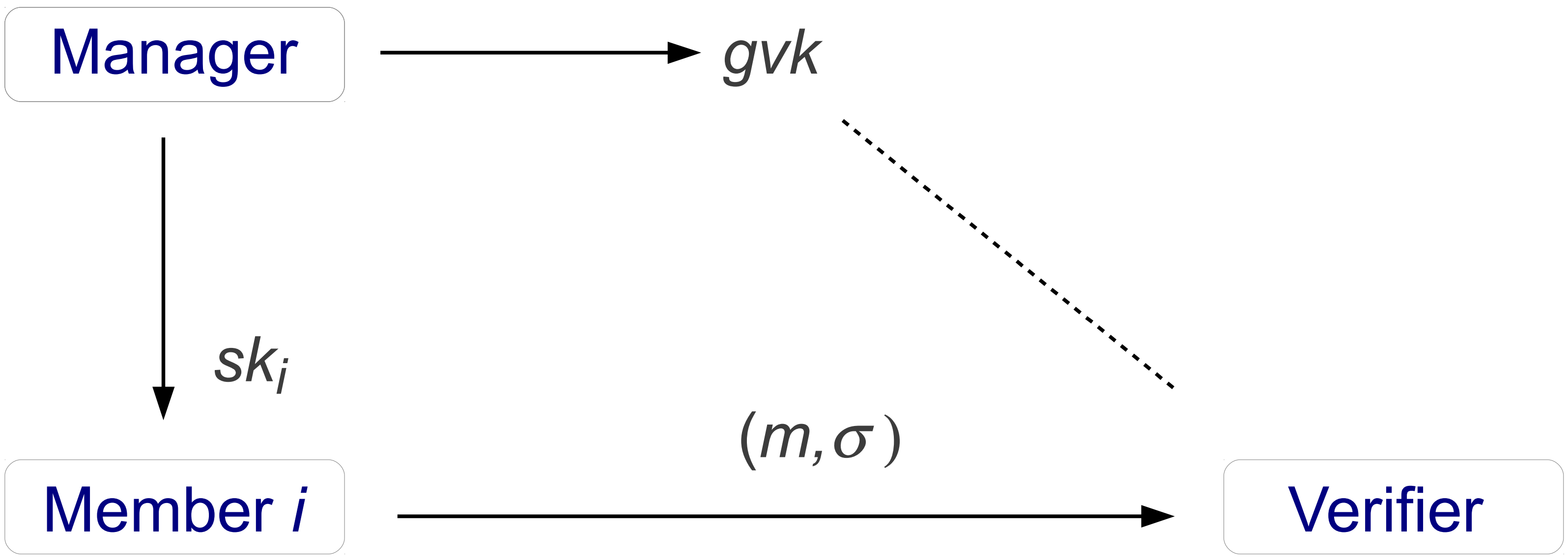
  Attribute-based signatures [MPR11]

  Anonymous credentials [CL01,BCKL08]

Group signatures

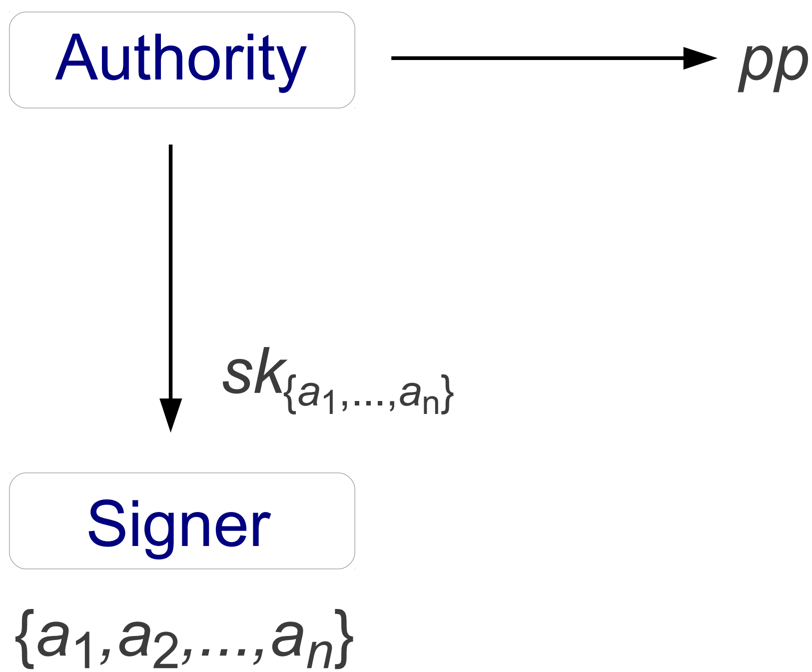


Group signatures



σ anonymous, but can be opened:

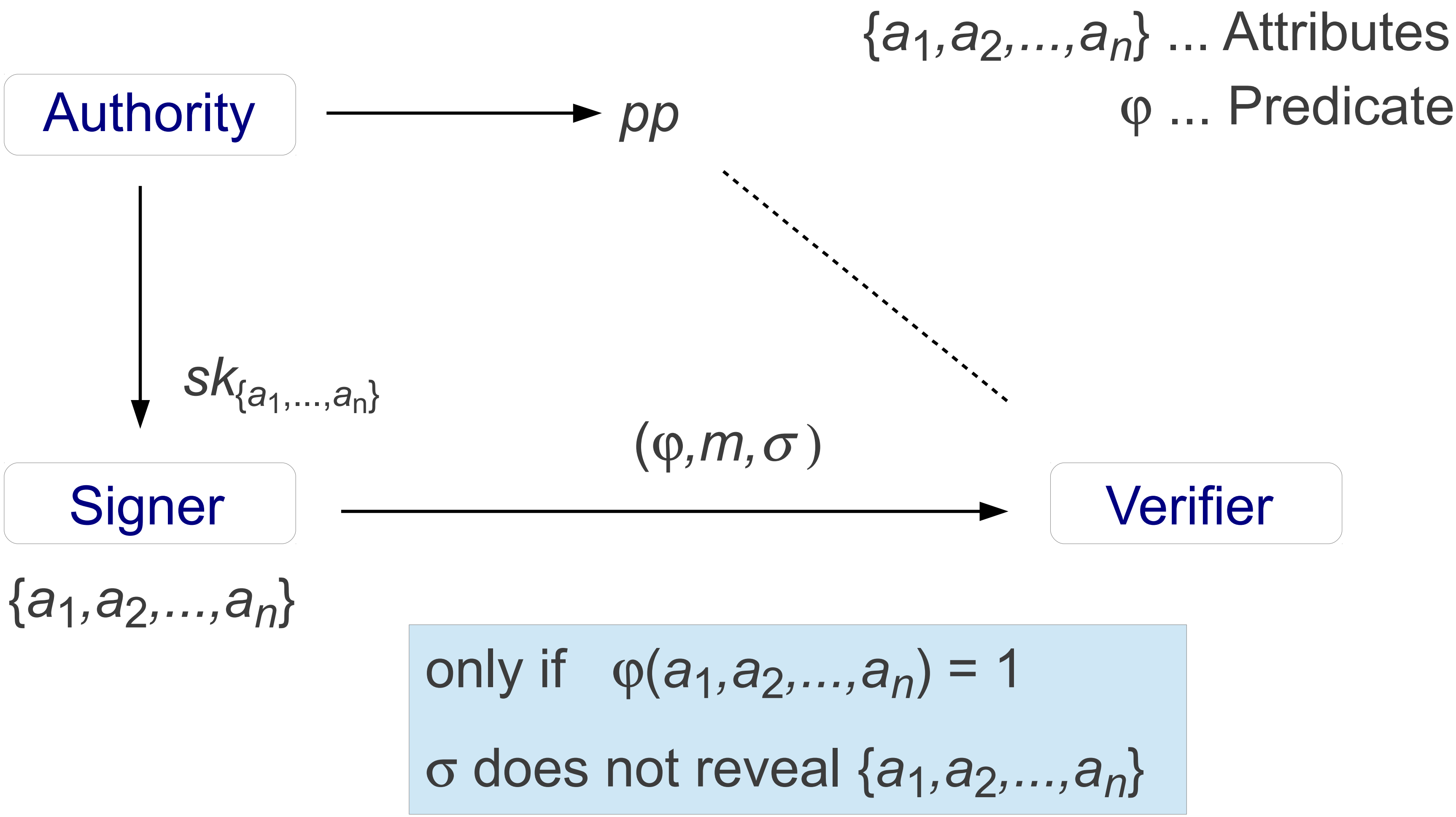
Attribute-based signatures



$\{a_1, a_2, \ldots, a_n\}$ ... Attributes

φ ... Predicate

Attribute-based signatures



Practical Motivation

- Company with public key *vk*
- Employees get signing keys enabling signing
  <u>anonymously</u> <u>on behalf of company</u>

Practical Motivation

- Company with public key *vk*
- Employees get signing keys enabling signing
  anonymously on behalf of company

- Group signatures:
  - Anonymous signing, no control of what can be signed

Practical Motivation

- Company with public key *vk*
- Employees get signing keys enabling signing
  anonymously on behalf of company

- Group signatures:
  - Anonymous signing, no control of what can be signed
- Attribute-based signatures:
  - Signing w.r.t policies like

$$\text{CEO} \vee (\text{board member} \wedge \text{manager})$$

Can we do better?

⇒ Public policies...

- Does verifier need to know?

Can we do better?

$\Rightarrow$ Public policies...

- Does verifier need to know?

$\Rightarrow$ Verification w.r.t. policies...

- Verifier must judge if message ok under policy:

$$\text{CEO} \vee \text{Intern}$$

Can we do better?

$\Rightarrow$ Public policies...

- Does verifier need to know?

$\Rightarrow$ Verification w.r.t. policies...

- Verifier must judge if message ok under policy:

CEO $\vee$ Intern

Policy-based signatures:

- No public policies
- Verification w.r.t. *vk* only

Can we do better?

⇒ Public policies...

- Does verifier need to know?

⇒ Verification w.r.t. policies...

- Verifier must judge if message ok under policy:

CEO $\vee$ Intern

Policy-based signatures:

- No public policies
- Verification w.r.t. *vk* only

Example:

Employee gets key with which she can sign contracts only with company *B.*

Definition of PBS

Definition

- Policy languages:

  We allow any language in **NP**, defined by policy checker

$$\mathrm{PC} : ((p,m),w) \rightarrow \{0,1\}$$

$$(p,m) \in L(\mathrm{PC}) \;:\Leftrightarrow\; \exists\, w \in \{0,1\}^* : \mathrm{PC}((p,m),w) = 1$$

... iff signing of *m* is permitted under *p*

Definition

- Policy languages:

  We allow any language in **NP**, defined by policy checker

  $$\mathrm{PC} : ((p,m),w) \to \{0,1\}$$

  $$(p,m) \in L(\mathrm{PC}) \ :\Leftrightarrow\ \exists\, w \in \{0,1\}^* : \mathrm{PC}((p,m),w) = 1$$

  ... iff signing of m is permitted under p

- Algorithms:

  $\mathrm{Setup}(1^\lambda) \to (pp,msk)$

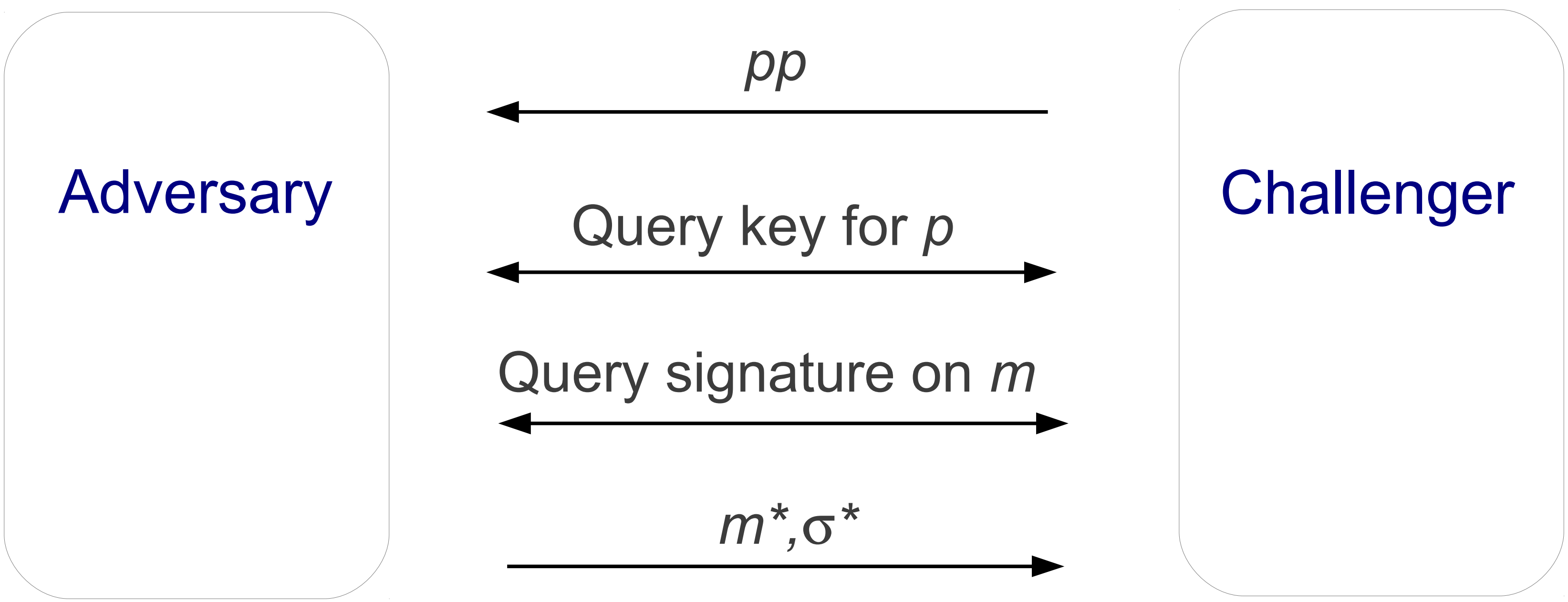
  $\mathrm{KeyGen}(msk,p) \to sk_p$

  $\mathrm{Sign}(sk_p,m,w) \to \sigma$

  $\mathrm{Verify}(pp,m,\sigma) \to b$

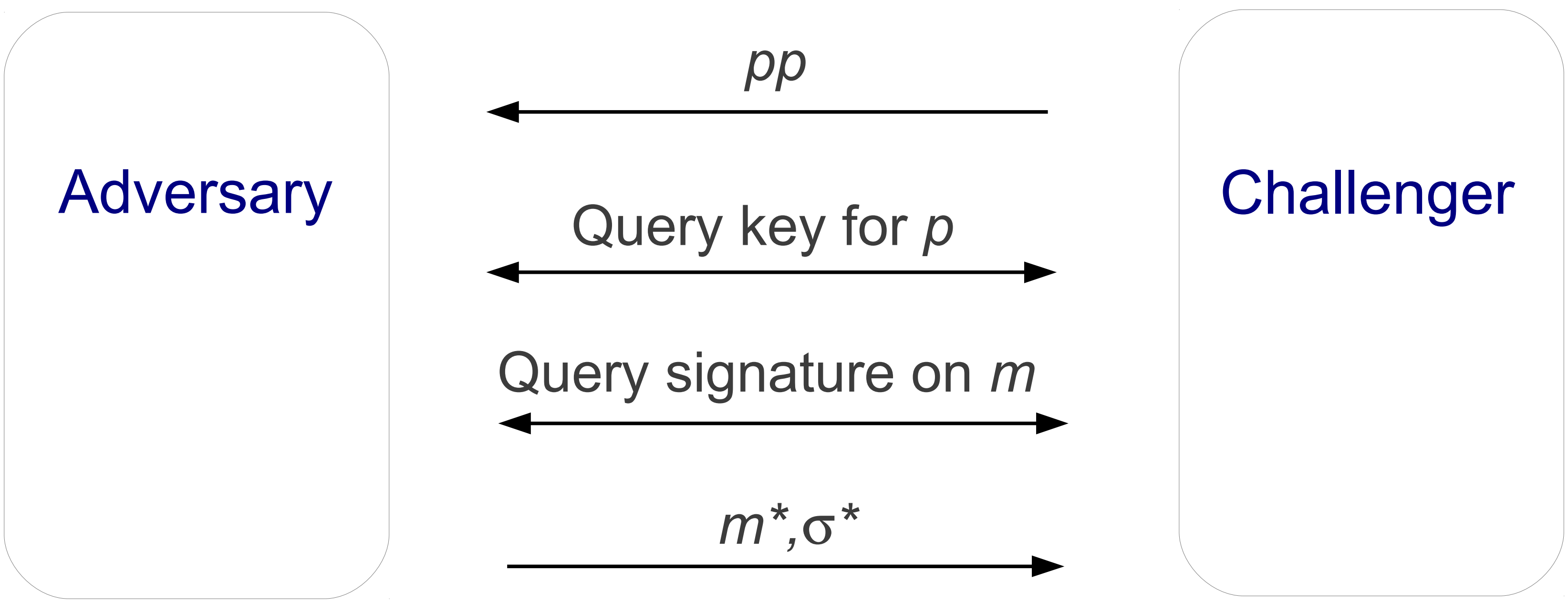
Unforgeability

- Game



Unforgeability

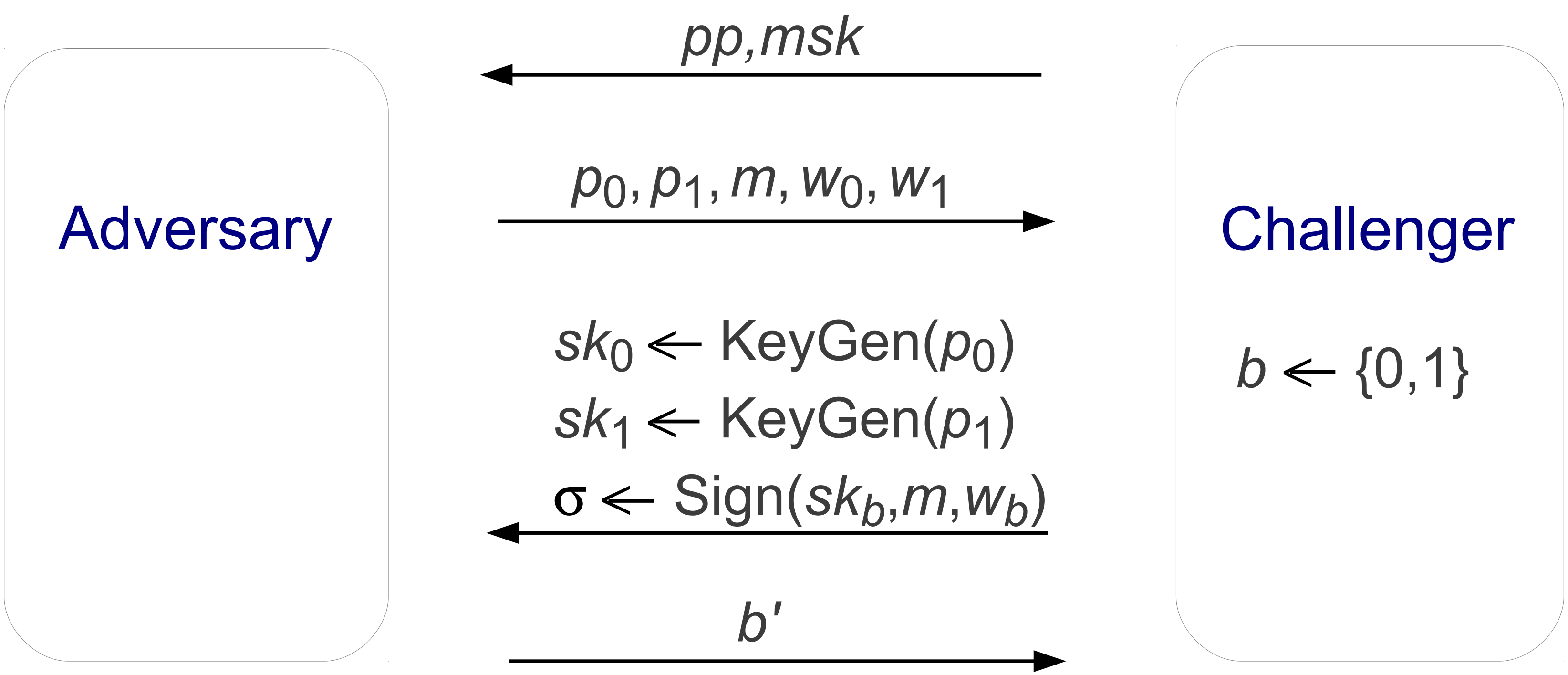
- Game



A wins if - Verify(pp, m^*, σ^*) = 1,
- no signature query for m^*,
- for all key queries for p: $(p, m^*) \notin L(\mathrm{PC})$

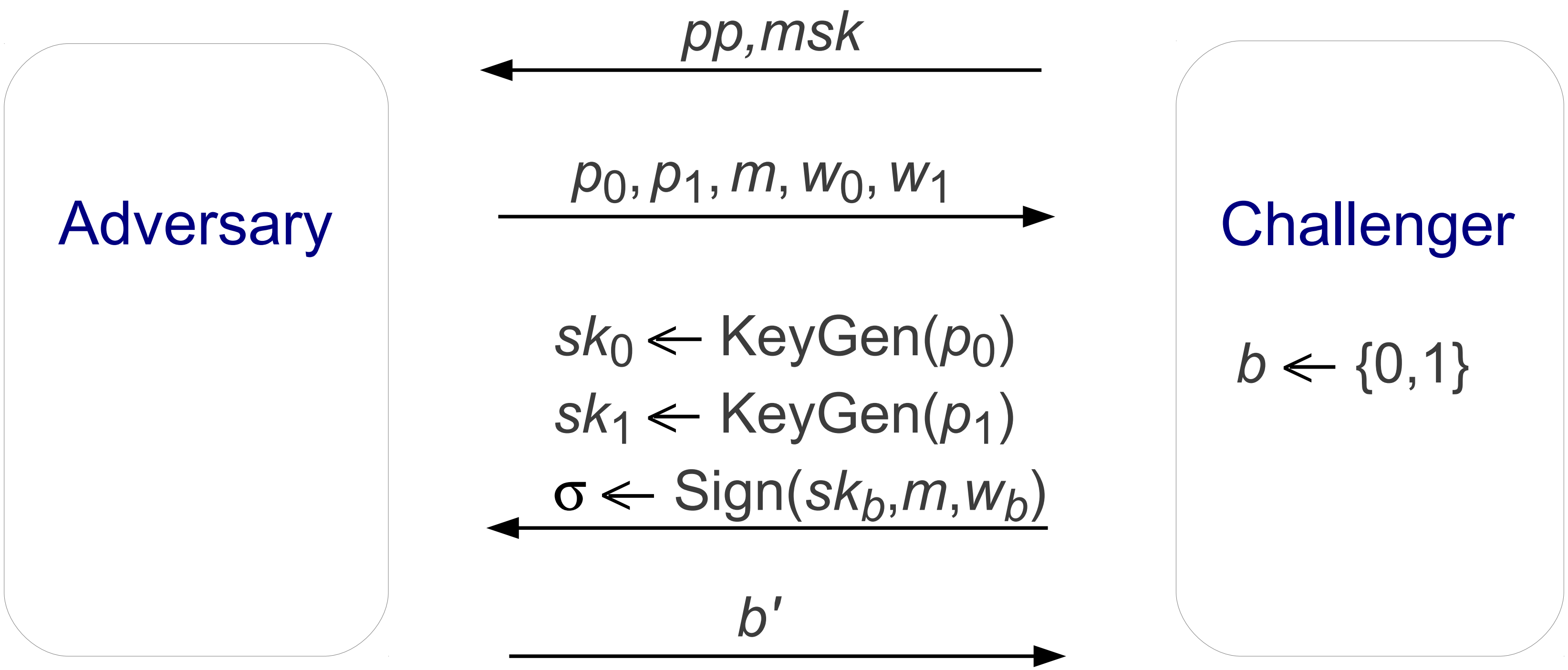
Indistinguishability

- Game



Indistinguishability

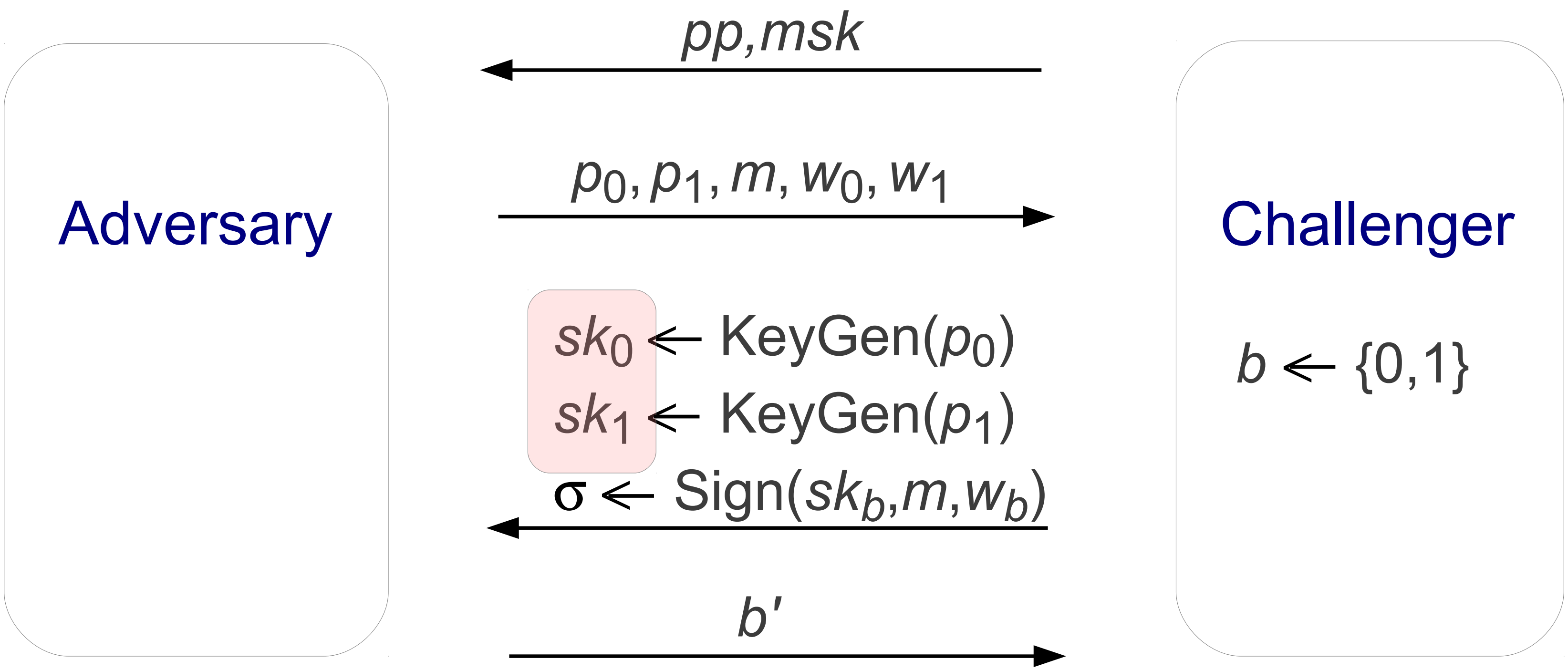
- Game



A wins if
- $\text{PC}((p_0,m),w_0) = \text{PC}((p_1,m),w_1) = 1$,
- $b' = b$

Indistinguishability

- Game



A wins if - $PC((p_0,m),w_0) = PC((p_1,m),w_1) = 1$,

- $b' = b$

Security notions

- Indistinguishability
  - Adversary gets sk_0 and sk_1 $\Rightarrow$ unlinkability

Security notions

- Indistinguishability
  - Adversary gets sk_0 and sk_1 $\Rightarrow$ unlinkability
  - Policy-revealing schemes still secure!
    (e.g. if only one policy per message)

Security notions

- Indistinguishability
  - Adversary gets sk_0 and sk_1 $\Rightarrow$ unlinkability
  - Policy-revealing schemes still secure!
    (e.g. if only one policy per message)

$\Rightarrow$ Simulation-based definition

Security notions

- Indistinguishability
  - Adversary gets sk_0 and sk_1 $\Rightarrow$ unlinkability
  - Policy-revealing schemes still secure!
    (e.g. if only one policy per message)

  $\Rightarrow$ Simulation-based definition

- Unforgeability
  - Not efficiently verifiable if game was won
    (have to check whether $(m^*, p) \notin L(\mathrm{PC})$)

Security notions

- Indistinguishability
  - Adversary gets sk_0 and sk_1 $\Rightarrow$ unlinkability
  - Policy-revealing schemes still secure!
    (e.g. if only one policy per message)

  $\Rightarrow$ Simulation-based definition

- Unforgeability
  - Not efficiently verifiable if game was won
    (have to check whether $(m^*, p) \notin L(\mathrm{PC})$)

  $\Rightarrow$ Extraction-based definition

Extr+Sim $\Rightarrow$ UF+IND

Constructions of PBS

Constructions

- Generic construction (à la [BMW03])

    based on - signatures

    - IND-CPA encryption
    - NIZK proofs

    for any policy language in **NP**

Constructions

- Generic construction (à la [BMW03])

  based on - signatures
  - IND-CPA encryption
  - NIZK proofs

  or based on
  - SSE-NIZK

  for any policy language in **NP**

Constructions

- Generic construction (à la [BMW03])

  based on - signatures
  - IND-CPA encryption
  - NIZK proofs

  or based on
  - SSE-NIZK

  for any policy language in **NP**

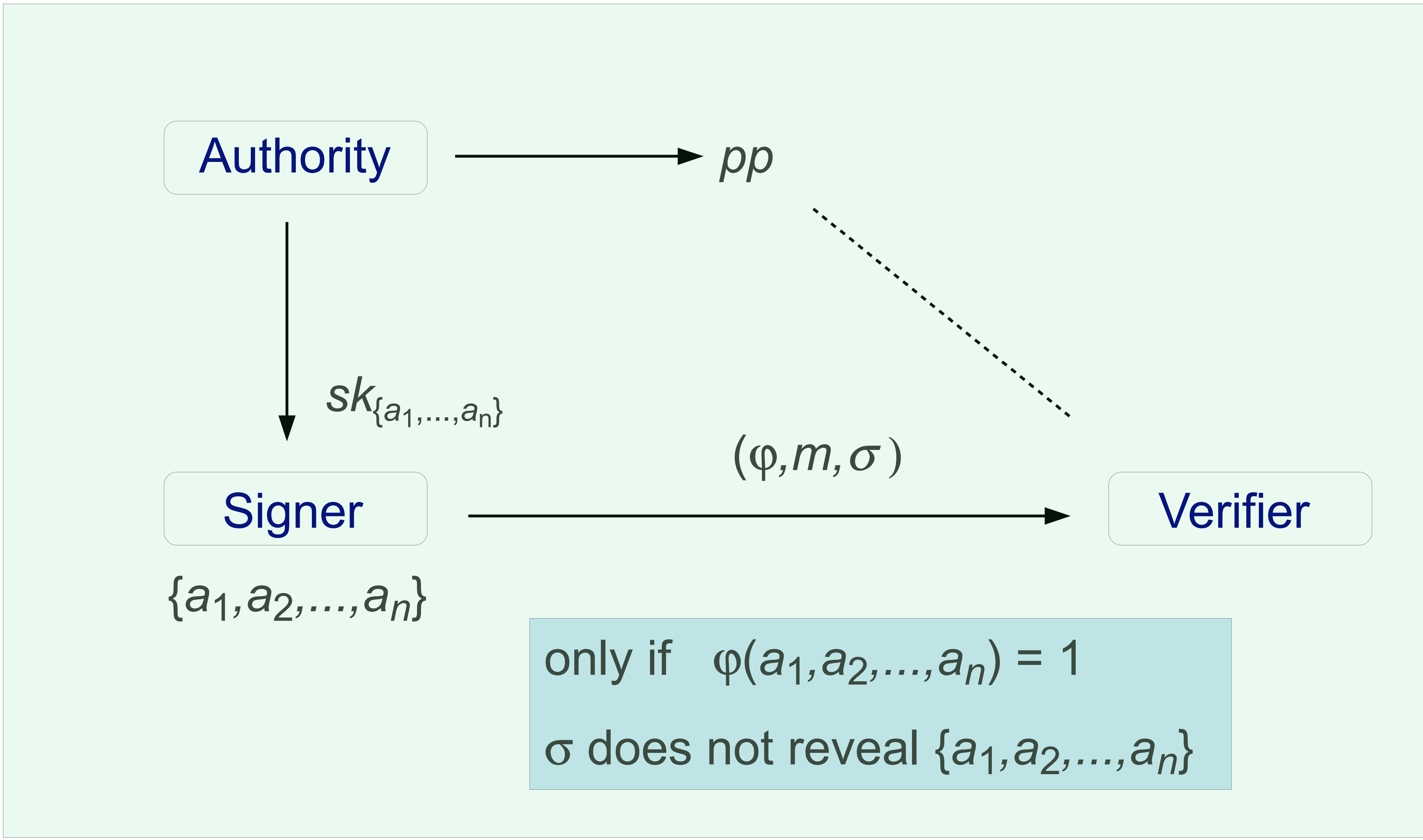
- Efficient construction

  based on - structure-preserving signatures [AFG+10]
  - Groth-Sahai proofs [GS08]

  for policy languages over **pairing groups**
  (policies define pairing-product equations)

Primitives from PBS

Attribute-based signatures from PBS



Attribute-based signatures from PBS

- Policies p ... set of attributes $A = \{a_1,a_2,...,a_n\}$
- PBS messages of form $M = (\varphi,m)$

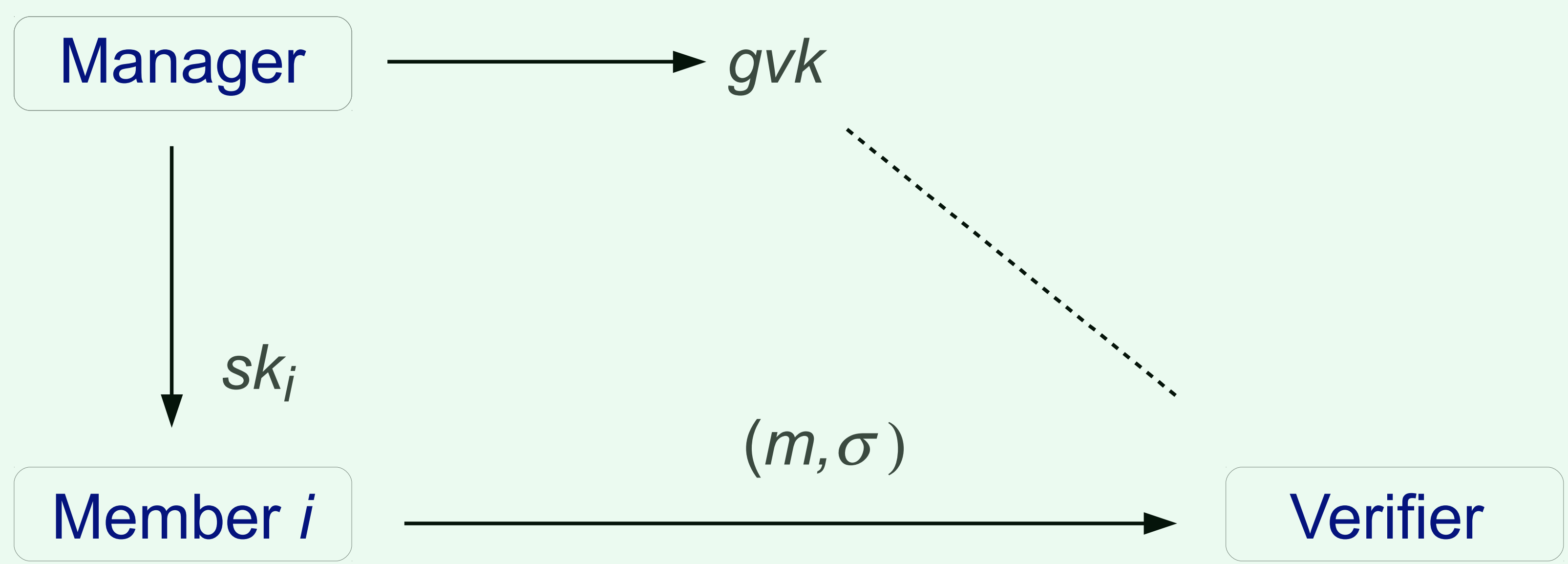
Attribute-based signatures from PBS

- Policies p ... set of attributes $A = \{a_1, a_2, ..., a_n\}$
- PBS messages of form $M = (\varphi, m)$
- PC : $(A, (\varphi, m))$

Attribute-based signatures from PBS

- Policies p ... set of attributes $A = \{a_1, a_2, ..., a_n\}$
- PBS messages of form $M = (\varphi, m)$
- PC : $(A, (\varphi, m)) \rightarrow \varphi(A)$

Group signatures from PBS



σ anonymous, but can be opened:

$$\sigma \xrightarrow{\text{Manager}} i$$

Group signatures from PBS

- Use public-key encryption scheme
- Policies *p* ... group-member identity *i*

Group signatures from PBS

- Use public-key encryption scheme
- Policies p ... group-member identity i
- PBS messages of form $M = (c,m)$
- PC : $\big(((ek,i),(c,m)),r\big) \rightarrow \big[c = \mathrm{Enc}(ek,i;r)\big]$

Group signatures from PBS

- Use public-key encryption scheme
- Policies p ... group-member identity i
- PBS messages of form $M = (c,m)$
- PC : $\big(((ek,i),(c,m)),r\big) \rightarrow [c = \mathrm{Enc}(ek,i;r)]$

GroupKeyGen: - create (ek,dk) for Enc, (pp,msk) for PBS

- member i gets key for $p = (ek,i)$

Sign(sk_i,m): encrypt i as c, sign (c,m), output $\Sigma = (c,\sigma)$

Group signatures from PBS

- Use public-key encryption scheme
- Policies p ... group-member identity i
- PBS messages of form $M = (c,m)$
- PC : $\big(((ek,i),(c,m)),r\big) \rightarrow [c = \mathrm{Enc}(ek,i;r)]$

GroupKeyGen: - create (ek,dk) for Enc, (pp,msk) for PBS

- member i gets key for $p = (ek,i)$

Sign(sk_i,m): encrypt i as c, sign (c,m), output $\Sigma = (c,\sigma)$

Verify: verify PBS Open$(dk,(c,\sigma))$: decrypt c

Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]

Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]
- CPA-secure public-key encryption

Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]
- CPA-secure public-key encryption

- combining the above [Sah99]: CCA-secure encryption

Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]
- CPA-secure public-key encryption

- combining the above [Sah99]: CCA-secure encryption

  thus PBS $\Rightarrow$ group signatures

Other primitives from PBS

- Simulation-sound extractable NIZK proofs [Gro06]
- CPA-secure public-key encryption

- combining the above [Sah99]: CCA-secure encryption

  thus PBS $\Rightarrow$ group signatures

- Signatures of knowledge [CL06]

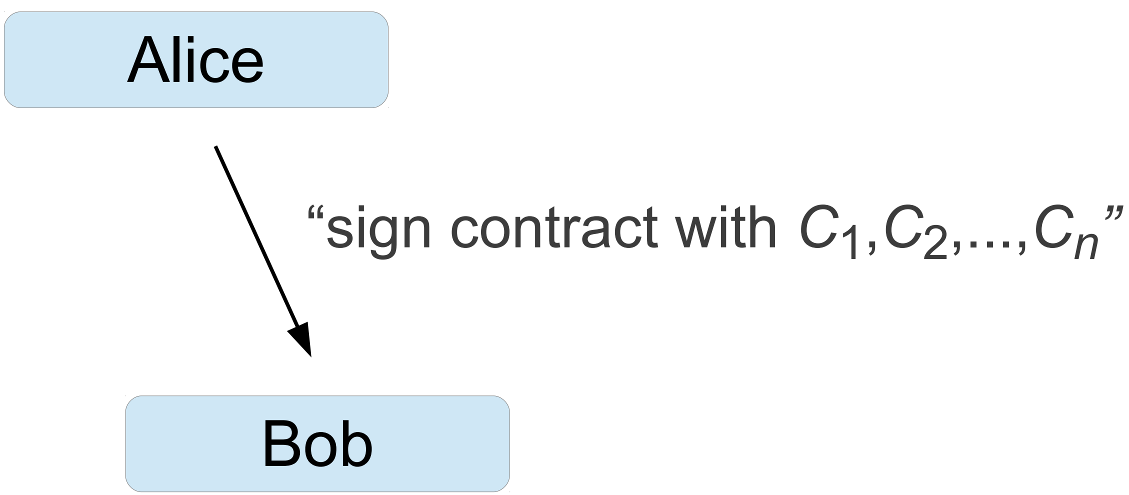
Delegatable PBS

Re-delegation

- Delegatable PBS
  - holding sk_p, one can delegate $sk_{p'}$ for subpolicy p'
- Reflects hierarchies in organizations

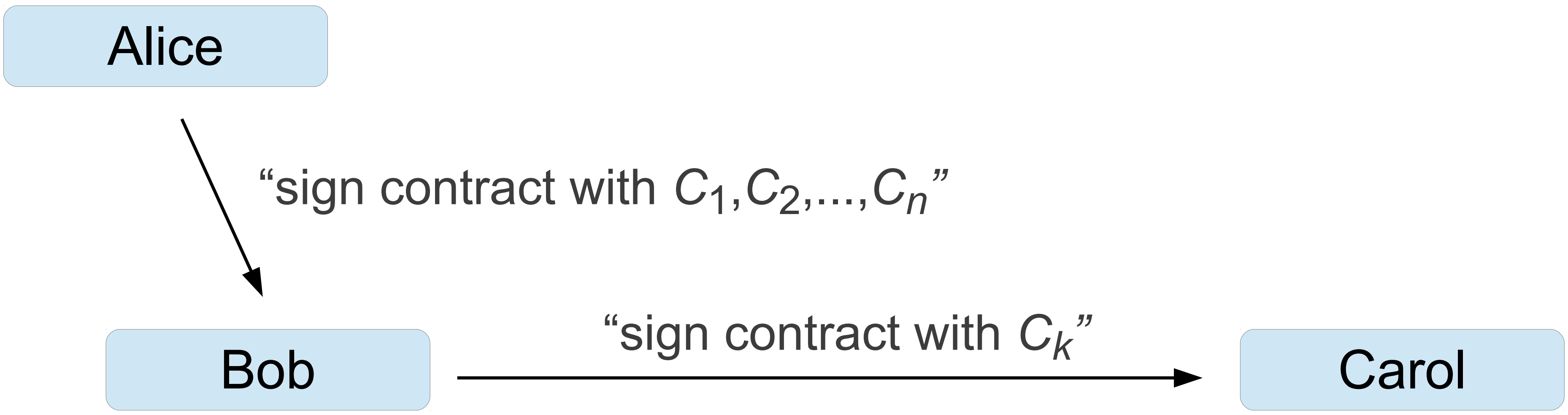
Re-delegation

- Delegatable PBS
  - holding sk_p, one can delegate $sk_{p'}$ for subpolicy p'
- Reflects hierarchies in organizations



Re-delegation

- Delegatable PBS
  - holding sk_p, one can delegate $sk_{p'}$ for subpolicy p'
- Reflects hierarchies in organizations



Thank you

PBS from SSE-NIZK

- Simulation-sound extractable NIZK:
  - prove membership for **NP** languages

- Authority has signature key pair (vk,sk)
- sk_p is signature on p
- PBS-signature on m is SSE-NIZK proof that $(vk,m) \in L$ defined by

$$((vk,m),(p,sig,w)) \in R_L \quad \Leftrightarrow$$

$$((p,m),w) \in \mathrm{PC} \wedge \mathrm{Verify}(vk,p,sig) = 1$$