Tracing a Linear Subspace

Application to Linearly-Homomorphic Group Signatures

Chloé Hébant[1] David Pointcheval[2] **Robert Schädlich**[2]

May 8, 2023

[1] Cosmian, Paris, France

[2] DIENS, École normale supérieure, PSL University, CNRS, Inria, Paris, France

Outline

1 Linearly-Homomorphic Group Signatures
- Definition
- Construction

2 A Core Technique: Tracing Linear Subspaces
- Trivial Solution
- Improved Efficiency via Code-Based Construction

Linearly-Homomorphic Group Signatures

Linearly-Homomorphic Group Signatures

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \to \sigma_i$

$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$

$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$

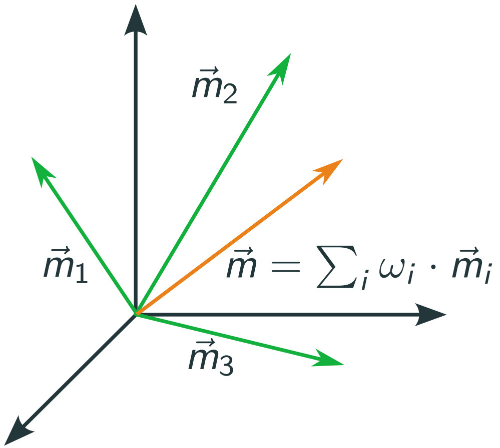
Linearly-Homomorphic Group Signatures

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \to \sigma_i$

$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$

$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$



$\boxed{\text{Linearly-Homomorphic}}$ Group Signatures

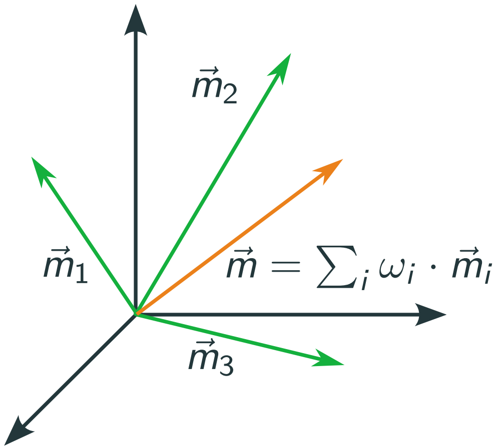
$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \to \sigma_i$

$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$

$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$

Security. EUF-CMA without trivial attacks

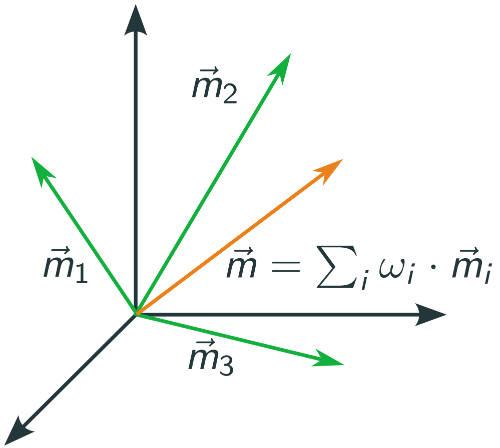


$\boxed{\text{Linearly-Homomorphic}}$ $\boxed{\text{Group}}$ Signatures

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$
$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \to \sigma_i$
$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$
$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$

$\mathsf{GKeyGen}(1^\lambda) \to (\mathsf{gpk}, \mathsf{gmsk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$
$\mathsf{GSign}(\mathsf{sk}_i, m) \to \sigma$
$\mathsf{GVerify}(\mathsf{gpk}, m, \sigma) \to 0 \,/\, 1$
$\mathsf{Open}(\mathsf{gmsk}, m, \sigma) \to i \in [n]$

Security. EUF-CMA without trivial attacks

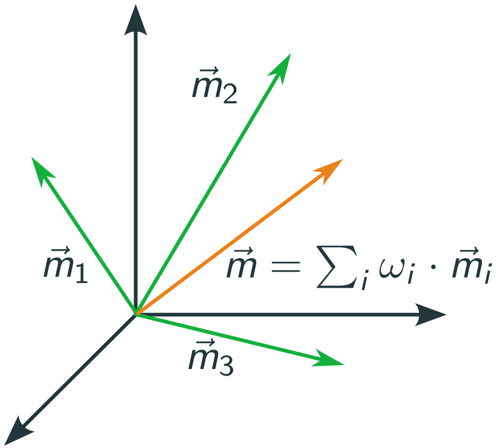


$\boxed{\text{Linearly-Homomorphic}}$ $\boxed{\text{Group}}$ Signatures

KeyGen$(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pk})$
Sign$(\mathsf{sk}, \vec{m}_i) \rightarrow \sigma_i$
Derive$(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \rightarrow \sigma$
Verify$(\mathsf{pk}, \vec{m}, \sigma) \rightarrow 0 \,/\, 1$

GKeyGen$(1^\lambda) \rightarrow (\mathsf{gpk}, \mathsf{gmsk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$
GSign$(\mathsf{sk}_i, m) \rightarrow \sigma$
GVerify$(\mathsf{gpk}, m, \sigma) \rightarrow 0 \,/\, 1$
Open$(\mathsf{gmsk}, m, \sigma) \rightarrow i \in [n]$

Security. EUF-CMA without trivial attacks



Linearly-Homomorphic Group Signatures

$\mathsf{KeyGen}(1^\lambda) \rightarrow (\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \rightarrow \sigma_i$

$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \rightarrow \sigma$

$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \rightarrow 0 \,/\, 1$

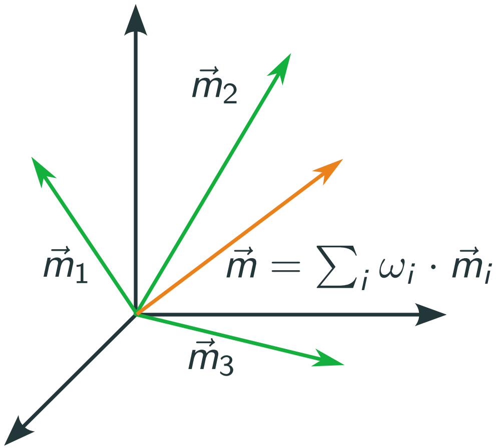
$\mathsf{GKeyGen}(1^\lambda) \rightarrow (\mathsf{gpk}, \mathsf{gmsk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$

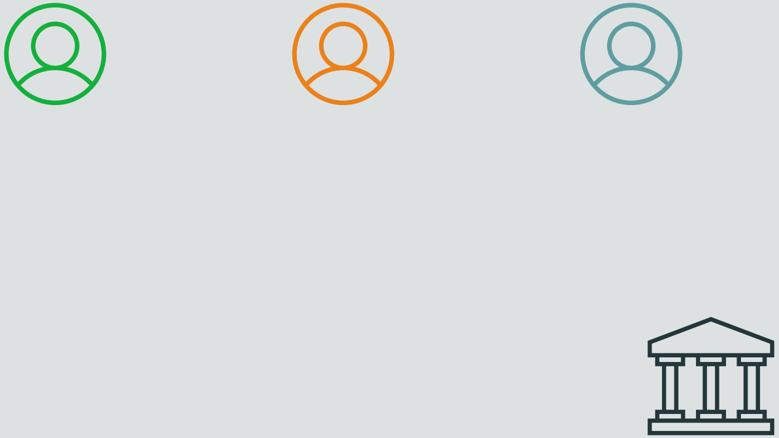
$\mathsf{GSign}(\mathsf{sk}_i, m) \rightarrow \sigma$

$\mathsf{GVerify}(\mathsf{gpk}, m, \sigma) \rightarrow 0 \,/\, 1$

$\mathsf{Open}(\mathsf{gmsk}, m, \sigma) \rightarrow i \in [n]$

Security. EUF-CMA without trivial attacks





Linearly-Homomorphic Group Signatures

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \to \sigma_i$

$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$

$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$

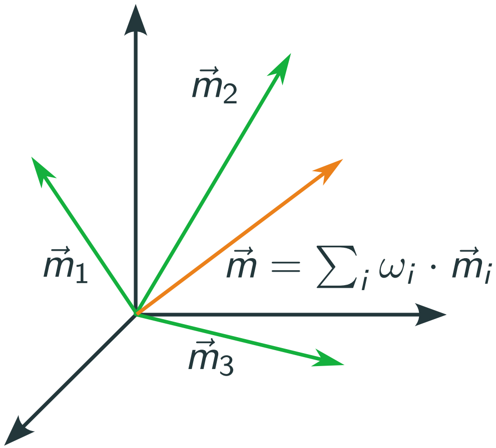
$\mathsf{GKeyGen}(1^\lambda) \to (\mathsf{gpk}, \mathsf{gmsk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$

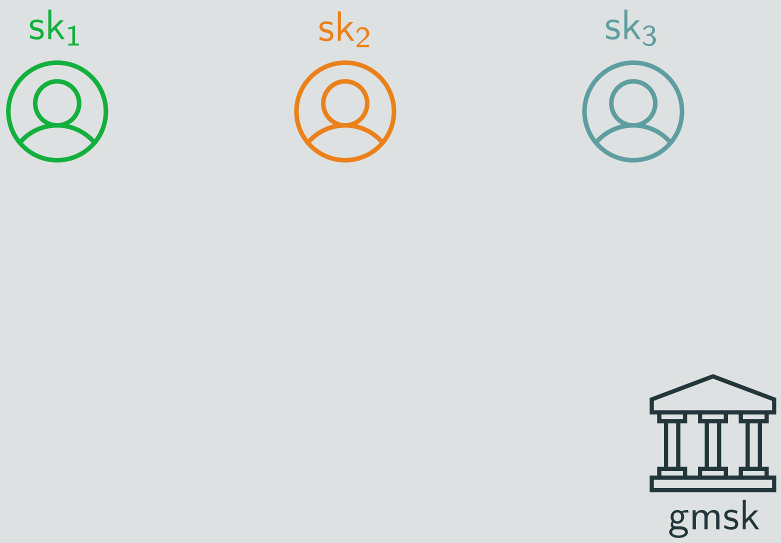
$\mathsf{GSign}(\mathsf{sk}_i, m) \to \sigma$

$\mathsf{GVerify}(\mathsf{gpk}, m, \sigma) \to 0 \,/\, 1$

$\mathsf{Open}(\mathsf{gmsk}, m, \sigma) \to i \in [n]$

Security. EUF-CMA without trivial attacks





Linearly-Homomorphic Group Signatures

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \to \sigma_i$

$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$

$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$

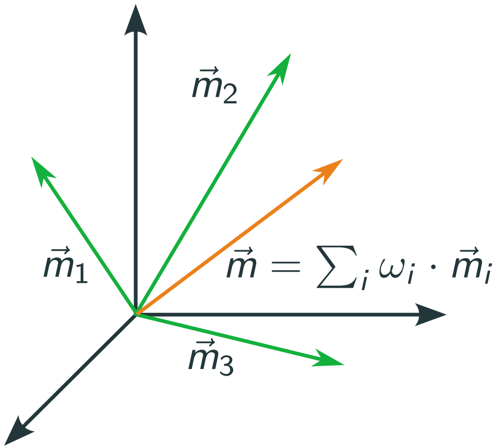
$\mathsf{GKeyGen}(1^\lambda) \to (\mathsf{gpk}, \mathsf{gmsk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$

$\mathsf{GSign}(\mathsf{sk}_i, m) \to \sigma$

$\mathsf{GVerify}(\mathsf{gpk}, m, \sigma) \to 0 \,/\, 1$

$\mathsf{Open}(\mathsf{gmsk}, m, \sigma) \to i \in [n]$

Security. EUF-CMA without trivial attacks



Linearly-Homomorphic Group Signatures

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$

$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \to \sigma_i$

$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$

$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$

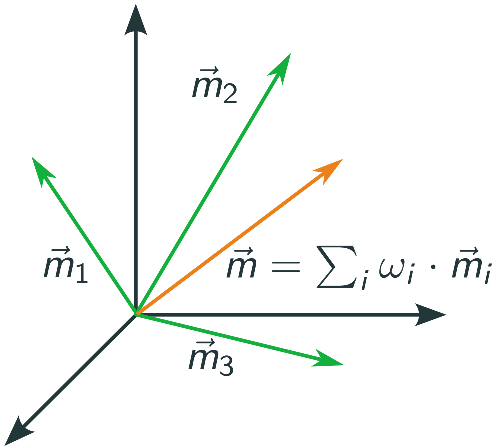
$\mathsf{GKeyGen}(1^\lambda) \to (\mathsf{gpk}, \mathsf{gmsk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$

$\mathsf{GSign}(\mathsf{sk}_i, m) \to \sigma$

$\mathsf{GVerify}(\mathsf{gpk}, m, \sigma) \to 0 \,/\, 1$

$\mathsf{Open}(\mathsf{gmsk}, m, \sigma) \to i \in [n]$

Security. EUF-CMA without trivial attacks

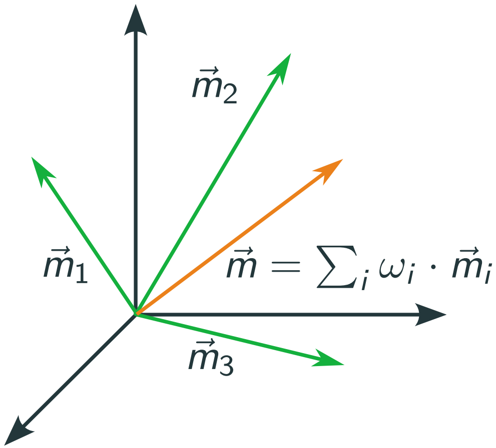


$\boxed{\text{Linearly-Homomorphic}}$ $\boxed{\text{Group}}$ Signatures

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$
$\mathsf{Sign}(\mathsf{sk}, \vec{m}_i) \to \sigma_i$
$\mathsf{Derive}(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$
$\mathsf{Verify}(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$

$\mathsf{GKeyGen}(1^\lambda) \to (\mathsf{gpk}, \mathsf{gmsk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$
$\mathsf{GSign}(\mathsf{sk}_i, m) \to \sigma$
$\mathsf{GVerify}(\mathsf{gpk}, m, \sigma) \to 0 \,/\, 1$
$\mathsf{Open}(\mathsf{gmsk}, m, \sigma) \to i \in [n]$

Security. EUF-CMA without trivial attacks



Anonymity.

Linearly-Homomorphic Group Signatures

KeyGen$(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$

Sign$(\mathsf{sk}, \vec{m}_i) \to \sigma_i$

Derive$(\mathsf{pk}, (\omega_i, \vec{m}_i, \sigma_i)_i) \to \sigma$

Verify$(\mathsf{pk}, \vec{m}, \sigma) \to 0 \,/\, 1$

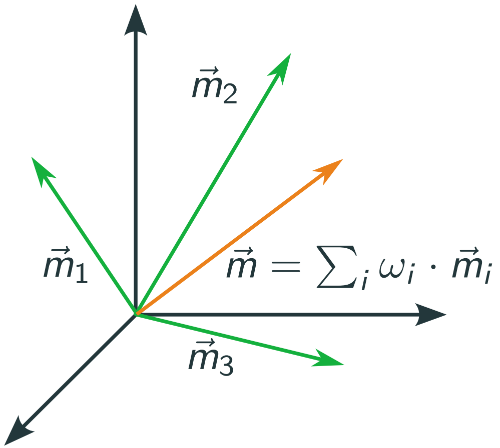
GKeyGen$(1^\lambda) \to (\mathsf{gpk}, \mathsf{gmsk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$

GSign$(\mathsf{sk}_i, m) \to \sigma$

GVerify$(\mathsf{gpk}, m, \sigma) \to 0 \,/\, 1$

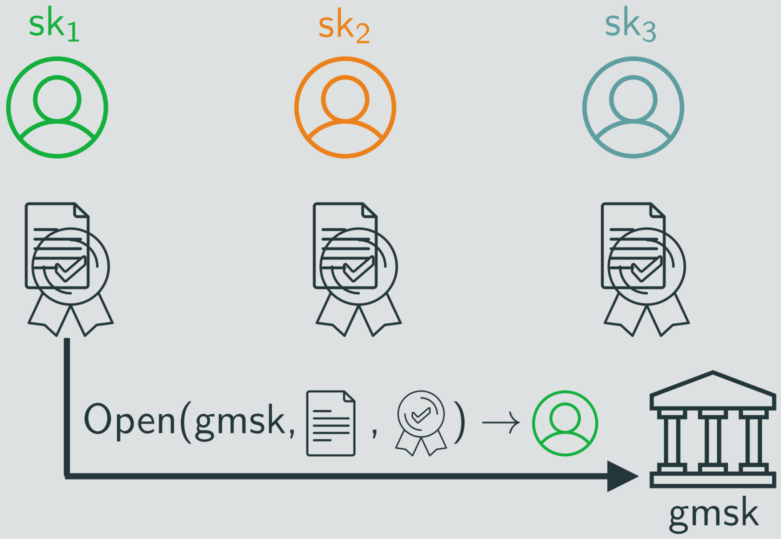
Open$(\mathsf{gmsk}, m, \sigma) \to i \in [n]$

Security. EUF-CMA without trivial attacks



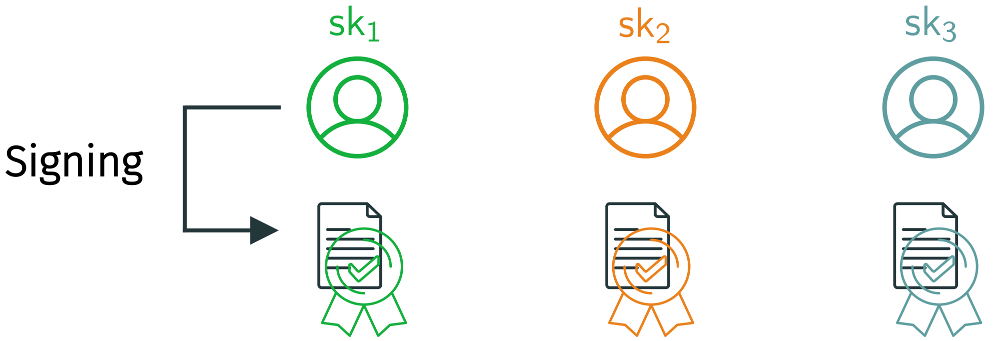
Anonymity.

Traceability.

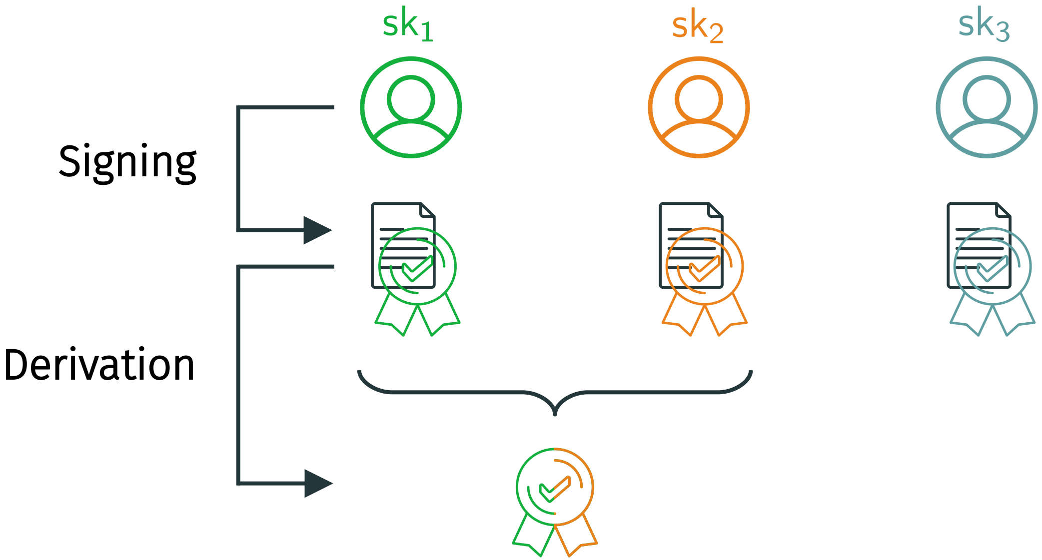


What Makes the Design Non-Trivial?

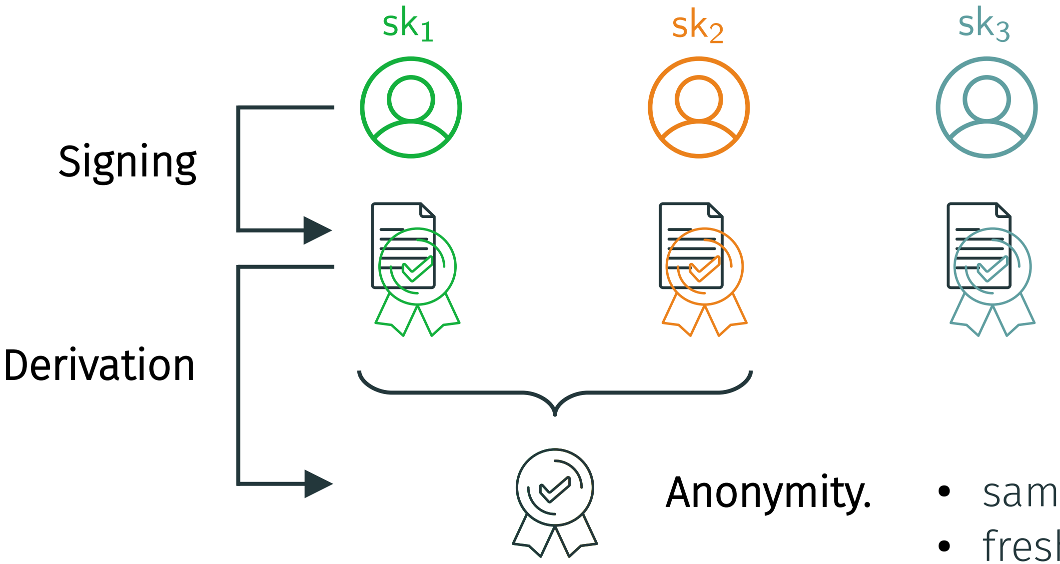
What Makes the Design Non-Trivial?



What Makes the Design Non-Trivial?



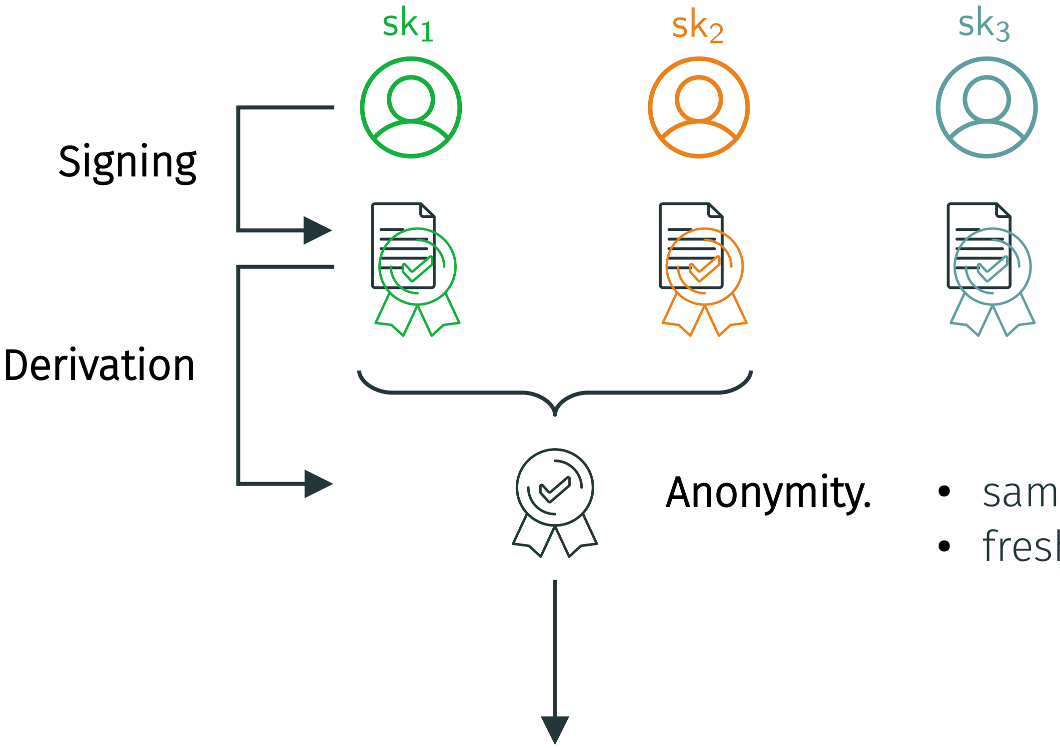
What Makes the Design Non-Trivial?



Anonymity.

- same user ⟷ different users
- freshly signed ⟷ previously derived

What Makes the Design Non-Trivial?



- same user ⟷ different users
- freshly signed ⟷ previously derived

Traceability.

A Framework for Group Signatures [BMW03]

A Framework for Group Signatures [BMW03]

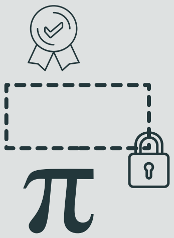
Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations

A Framework for Group Signatures [BMW03]

Ingredients.

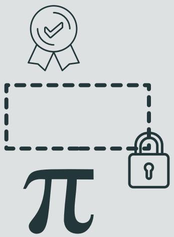
- (traditional) signature
- public-key encryption
- NIZK for NP relations



A Framework for Group Signatures [BMW03]

Ingredients.

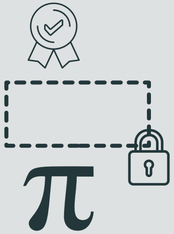
- (traditional) signature
- public-key encryption
- NIZK for NP relations



A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations



A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations

π

$\mathsf{sk}_1 = (\; , \;)$

$\mathsf{sk}_2 = (\; , \;)$

$\mathsf{sk}_3 = (\; , \;)$

$\mathsf{gmsk} = (\;)$

Signing. $\mathsf{GSign}(\mathsf{sk}_1 , \;)$

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations

π

Signing. GSign(sk_1 ,)

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations

π

$\mathsf{sk}_1 = (\; , \;)$

$\mathsf{sk}_2 = (\; , \;)$

$\mathsf{sk}_3 = (\; , \;)$

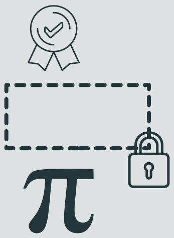
$\mathsf{gmsk} = (\;)$

Signing. GSign(sk_1 ,)

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations

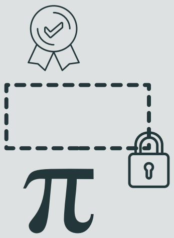


Signing. GSign(sk_1 , )

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations

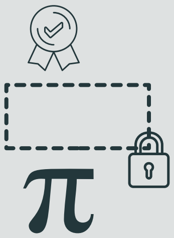


Signing. $\mathsf{GSign}(\ \mathsf{sk}_1\ ,$  $)$

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations



Signing. GSign(sk_1 ,)

Opening. Open(gmsk, , (, π))

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations π

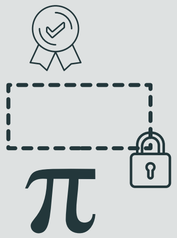
Signing. GSign(sk_1 ,)

Opening. Open(gmsk, , (, π))

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations



Signing. GSign(sk_1 ,)

Opening. Open(gmsk, , (, π))

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations π

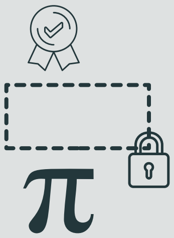
Signing. GSign(sk_1 ,)

Opening. Open(gmsk, , (, π))

A Framework for Group Signatures [BMW03]

Ingredients.

- (traditional) signature
- public-key encryption
- NIZK for NP relations



Signing. GSign(sk_1 ,   )

Opening. Open(gmsk,   , (, π))

How to Make This Linearly Homomorphic?

How to Make This Linearly Homomorphic?

How to Make This Linearly Homomorphic?

π

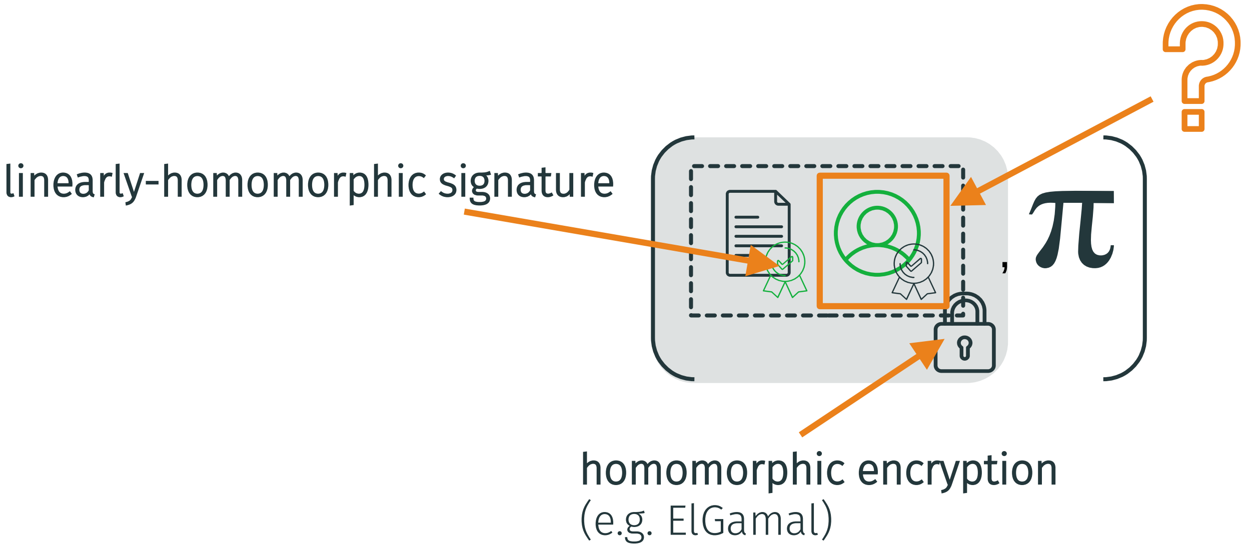
homomorphic encryption
(e.g. ElGamal)

How to Make This Linearly Homomorphic?

linearly-homomorphic signature

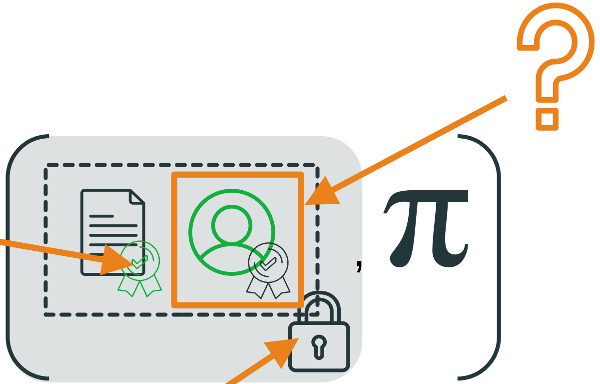
homomorphic encryption
(e.g. ElGamal)

How to Make This Linearly Homomorphic?



How to Make This Linearly Homomorphic?

linearly-homomorphic signature

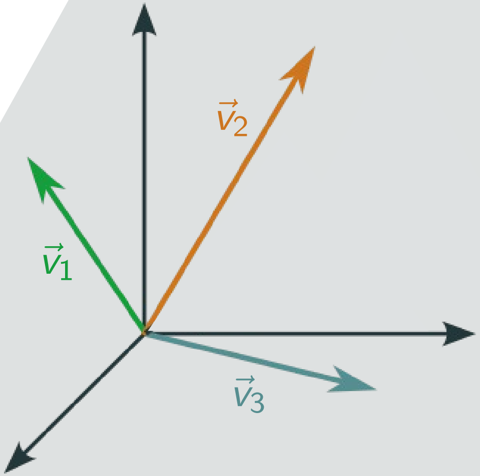


homomorphic encryption
(e.g. ElGamal)

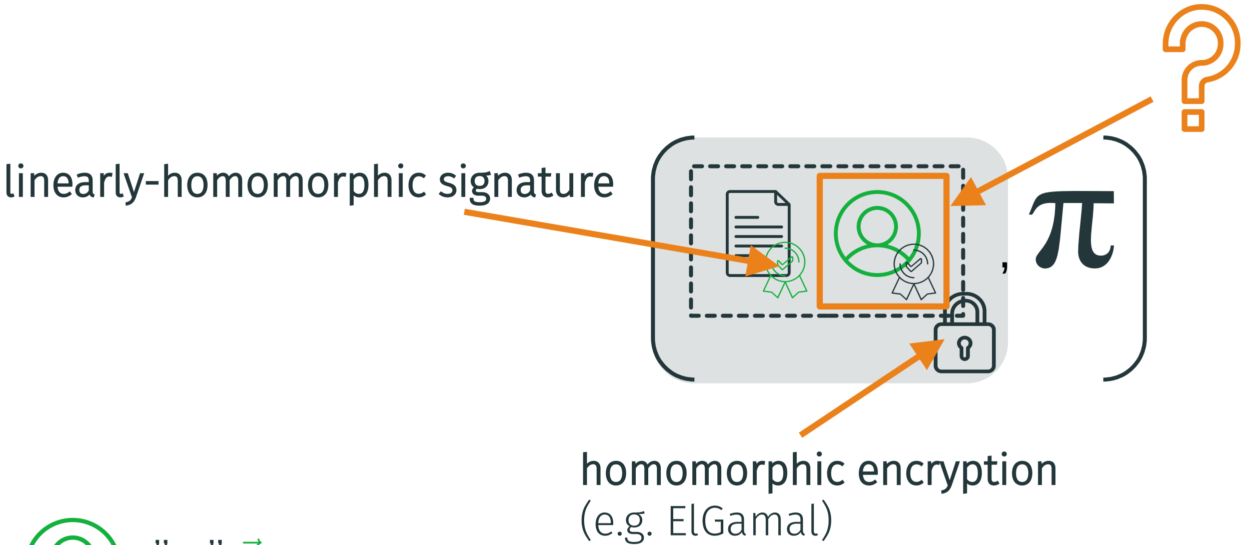
"=" $\vec{v}_1$

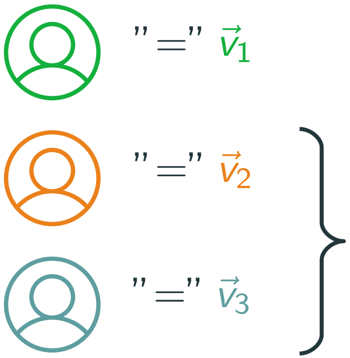
"=" $\vec{v}_2$

"=" $\vec{v}_3$



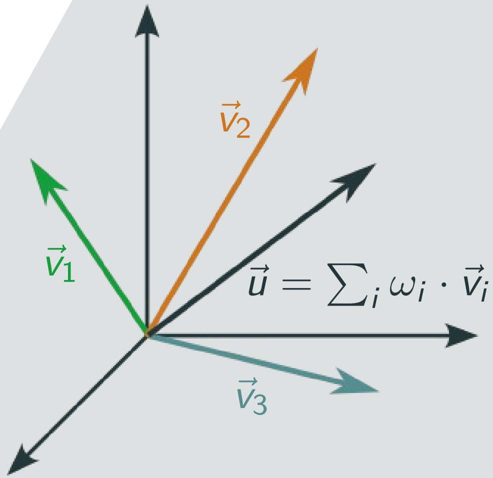
How to Make This Linearly Homomorphic?





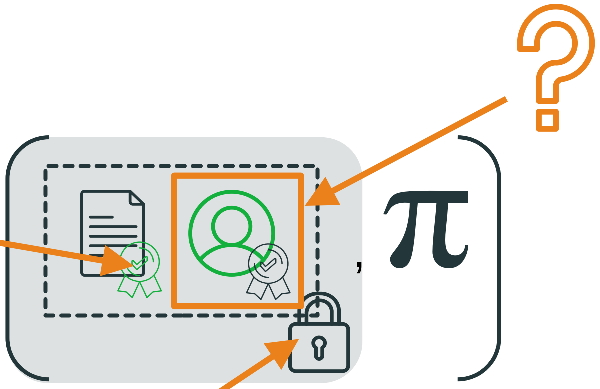
Signature derivation for and .

$$\vec{u} = \omega_1 \cdot \vec{v}_1 + \omega_2 \cdot \vec{v}_2 + \omega_3 \cdot \vec{v}_3$$



How to Make This Linearly Homomorphic?

linearly-homomorphic signature



homomorphic encryption
(e.g. ElGamal)

$" = " \vec{v}_1$

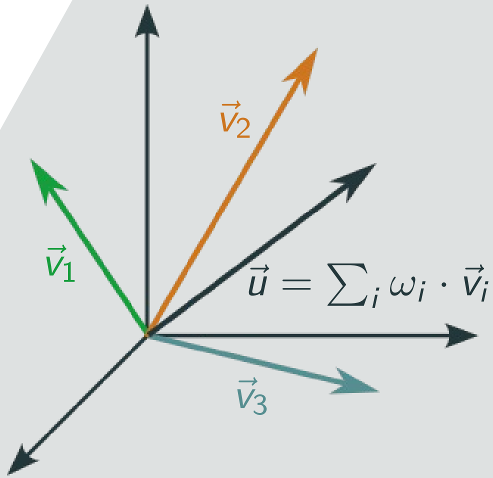
$" = " \vec{v}_2$

$" = " \vec{v}_3$

Signature derivation for and .

$$\vec{u} = \boxed{\omega_1} \cdot \vec{v}_1 + \omega_2 \cdot \vec{v}_2 + \omega_3 \cdot \vec{v}_3$$

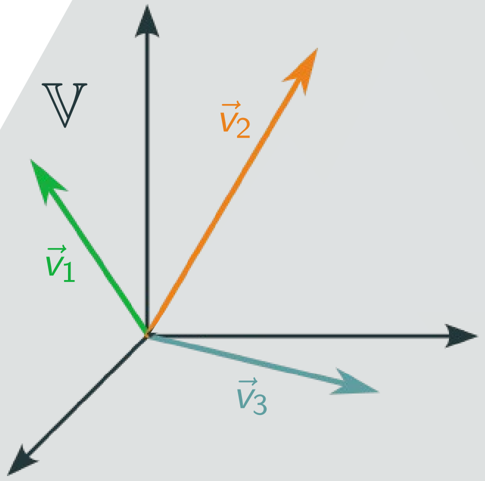
$$\omega_1 = 0$$



“Tracing” a Linear Subspace

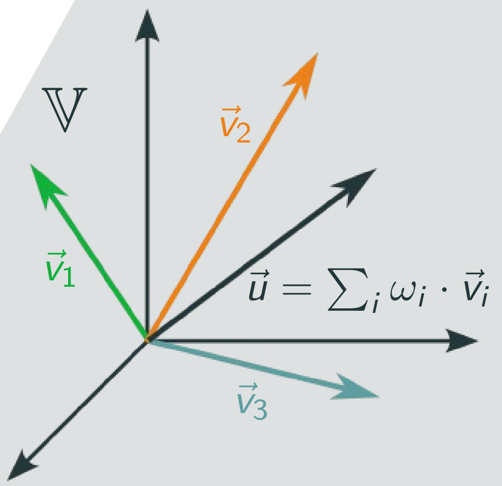
"Tracing" a Linear Subspace

Given: • basis $\mathcal{B} = \{\vec{v}_1, \ldots, \vec{v}_n\}$ of vector space $\mathbb{V} = \mathsf{span}(\mathcal{B})$



"Tracing" a Linear Subspace

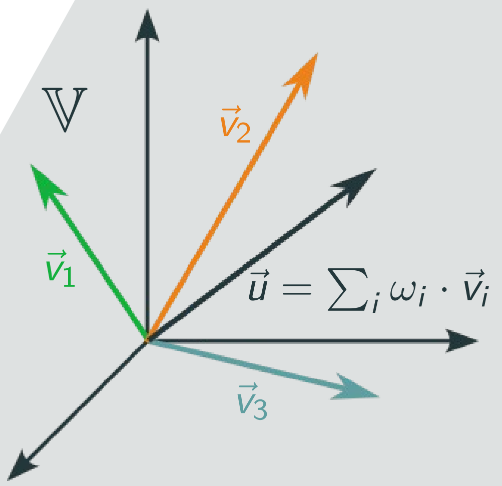
Given:
- basis $\mathcal{B} = \{\vec{v}_1, \dots, \vec{v}_n\}$ of vector space $\mathbb{V} = \mathsf{span}(\mathcal{B})$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathbb{V}$



“Tracing” a Linear Subspace

Given:
- basis $\mathcal{B} = \{\vec{v}_1, \dots, \vec{v}_n\}$ of vector space $\mathbb{V} = \mathrm{span}(\mathcal{B})$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathbb{V}$

Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$

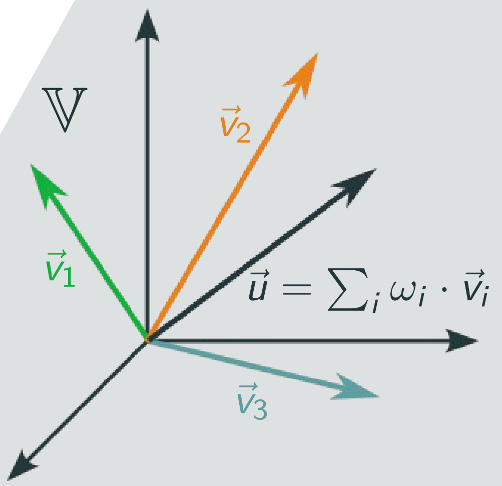


"Tracing" a Linear Subspace

Given:
- basis $\mathcal{B} = \{\vec{v}_1, \ldots, \vec{v}_n\}$ of vector space $\mathbb{V} = \mathsf{span}(\mathcal{B})$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathbb{V}$

Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$

Intuition: given $\vec{u} \in \mathsf{span}(\mathcal{B})$, find smallest set $\mathcal{X} \subseteq \mathcal{B}$ s.t. $\vec{u} \in \mathsf{span}(\mathcal{X})$



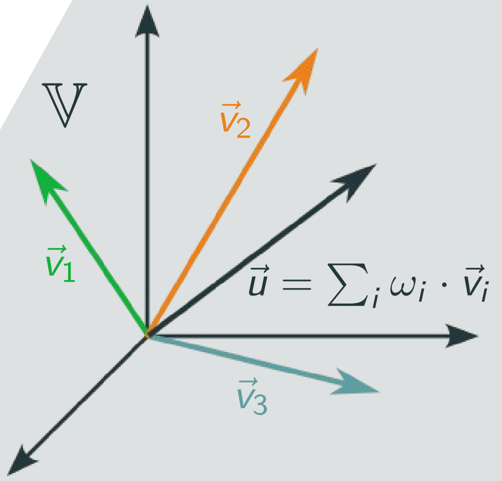
“Tracing” a Linear Subspace

Given:
- basis $\mathcal{B} = \{\vec{v}_1, \ldots, \vec{v}_n\}$ of vector space $\mathbb{V} = \mathsf{span}(\mathcal{B})$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathbb{V}$

Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$

Intuition: given $\vec{u} \in \mathsf{span}(\mathcal{B})$, find smallest set $\mathcal{X} \subseteq \mathcal{B}$ s.t. $\vec{u} \in \mathsf{span}(\mathcal{X})$

Isn’t that trivial?



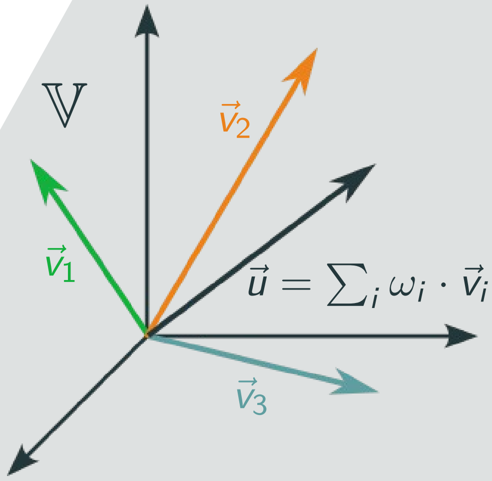
"Tracing" a Linear Subspace

Given:
- basis $\mathcal{B} = \{\vec{v}_1, \dots, \vec{v}_n\}$ of vector space $\mathbb{V} = \mathsf{span}(\mathcal{B})$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathbb{V}$

Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$

Intuition: given $\vec{u} \in \mathsf{span}(\mathcal{B})$, find smallest set $\mathcal{X} \subseteq \mathcal{B}$ s.t. $\vec{u} \in \mathsf{span}(\mathcal{X})$

Isn't that trivial? **YES!**



"Tracing" a Linear Subspace

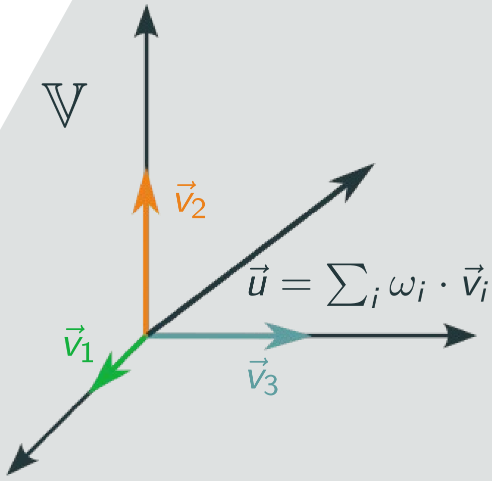
Given:
- basis $\mathcal{B} = \{\vec{v}_1, \dots, \vec{v}_n\}$ of vector space $\mathbb{V} = \mathsf{span}(\mathcal{B})$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathbb{V}$

Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$

Intuition: given $\vec{u} \in \mathsf{span}(\mathcal{B})$, find smallest set $\mathcal{X} \subseteq \mathcal{B}$ s.t. $\vec{u} \in \mathsf{span}(\mathcal{X})$

Isn't that trivial? **YES!**

E.g., choose $\mathbb{V} = \mathbb{Z}_p^n$ for a prime p and $\vec{v}_i = \vec{e}_i$ (i-th unit vector)
On input $\vec{u} = (u_1, \dots, u_n)$, return $\mathcal{I} = \{i : u_i \neq 0\}$



"Tracing" a Linear Subspace

Given:
- basis $\mathcal{B} = \{\vec{v}_1, \ldots, \vec{v}_n\}$ of vector space $\mathbb{V} = \mathsf{span}(\mathcal{B})$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathbb{V}$

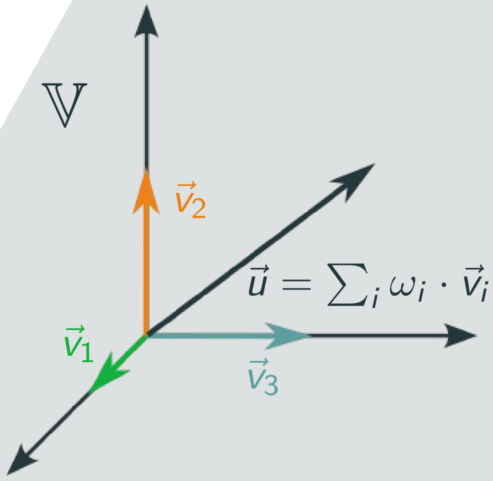
Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$

Intuition: given $\vec{u} \in \mathsf{span}(\mathcal{B})$, find smallest set $\mathcal{X} \subseteq \mathcal{B}$ s.t. $\vec{u} \in \mathsf{span}(\mathcal{X})$

Isn't that trivial? **YES!**

E.g., choose $\mathbb{V} = \mathbb{Z}_p^n$ for a prime p and $\vec{v}_i = \vec{e}_i$ (i-th unit vector)
On input $\vec{u} = (u_1, \ldots, u_n)$, return $\mathcal{I} = \{i : u_i \neq 0\}$

Note: trivial solution is *optimal* but *inefficient*



Tracing against Bounded Collusions

Relaxation of traceability: introduce an upper bound c on the maximum size of collusions

- never accuse honest user
- correct opening for collusions of size $\leq c$

Tracing against Bounded Collusions

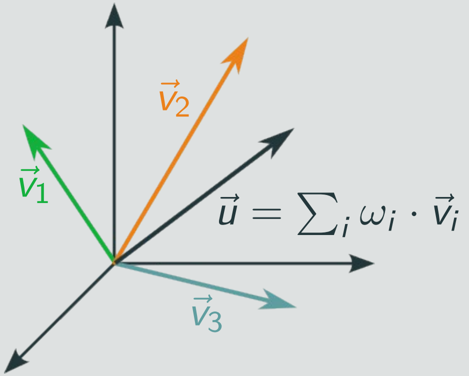
Relaxation of traceability: introduce an upper bound c on the maximum size of collusions

- never accuse honest user
- correct opening for collusions of size $\leq c$

What does that mean for the subspace-tracing problem?

Given:
- linearly independent vectors $\mathcal{B} = \{\vec{v}_1, \ldots, \vec{v}_n\}$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathrm{span}(\mathcal{B})$

Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$



Tracing against Bounded Collusions

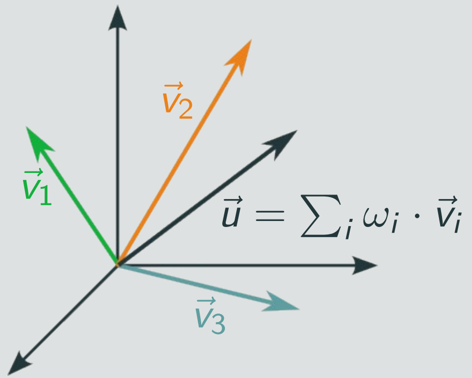
Relaxation of traceability: introduce an upper bound c on the maximum size of collusions

- never accuse honest user
- correct opening for collusions of size $\leq c$

What does that mean for the subspace-tracing problem?

Given:
- c-linearly independent vectors $\mathcal{B} = \{\vec{v}_1, \ldots, \vec{v}_n\}$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathrm{span}_c(\mathcal{B})$

Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$



Tracing against Bounded Collusions

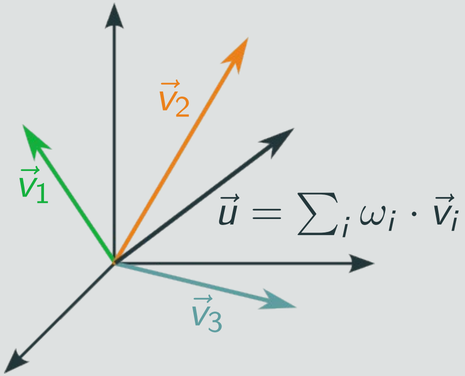
Relaxation of traceability: introduce an upper bound c on the maximum size of collusions

- never accuse honest user
- correct opening for collusions of size $\leq c$

What does that mean for the subspace-tracing problem?

Given:
- c-linearly independent vectors $\mathcal{B} = \{\vec{v}_1, \ldots, \vec{v}_n\}$
- $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i \in \mathrm{span}_c(\mathcal{B})$

Task: return $\mathcal{I} := \{i : \omega_i \neq 0\} \subseteq [n]$



Efficiency comparison:

- trivial solution $\mathcal{O}(n)$
- our work $\mathcal{O}(c^2 \cdot \log(n/\varepsilon))$ ($\varepsilon =$ maximum acceptable error probability)

Linear-Subspace Tracing: Under the Hood (if time permits...)

Analogy to IPP (and fingerprinting) codes:

- vectors $\longleftrightarrow$ codewords
- linear combinations $\longleftrightarrow$ descendants

Linear-Subspace Tracing: Under the Hood (if time permits...)

Analogy to IPP (and fingerprinting) codes:

- vectors $\longleftrightarrow$ codewords
- linear combinations $\longleftrightarrow$ descendants

vectors $\{\vec{v}_i : \omega_i \neq 0\} \subseteq \mathcal{B}$

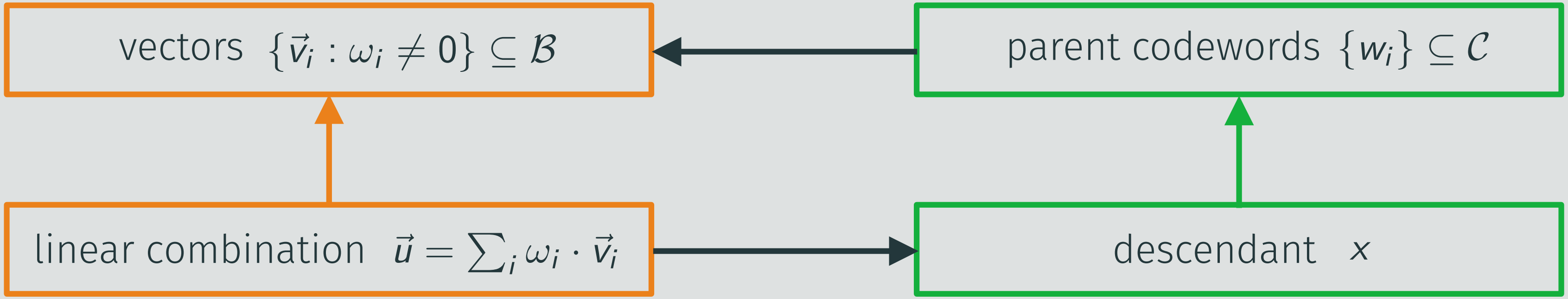
$\uparrow$

linear combination $\vec{u} = \sum_i \omega_i \cdot \vec{v}_i$

Linear-Subspace Tracing: Under the Hood (if time permits...)

Analogy to IPP (and fingerprinting) codes:

- vectors $\longleftrightarrow$ codewords
- linear combinations $\longleftrightarrow$ descendants

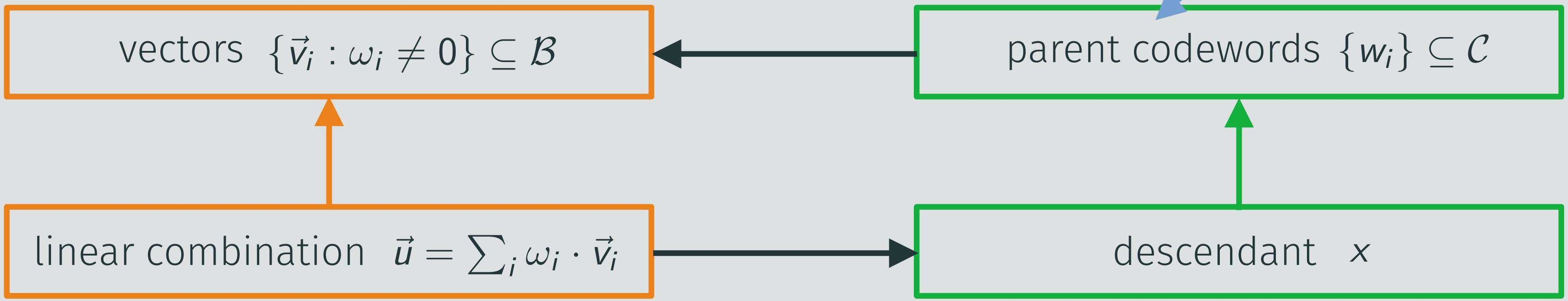


Linear-Subspace Tracing: Under the Hood (if time permits...)

Analogy to IPP (and fingerprinting) codes:

- vectors $\longleftrightarrow$ codewords
- linear combinations $\longleftrightarrow$ descendants

IPP codes trace only a *single* parent,
so we introduce *fully* IPP codes



Linear-Subspace Tracing: Under the Hood (if time permits...)

Analogy to IPP (and fingerprinting) codes:

- vectors $\longleftrightarrow$ codewords
- linear combinations $\longleftrightarrow$ descendants

IPP codes trace only a *single* parent,
so we introduce *fully* IPP codes

$$\begin{tikzcd}
\text{vectors } \{\vec{v}_i : \omega_i \neq 0\} \subseteq \mathcal{B} & \text{parent codewords } \{w_i\} \subseteq \mathcal{C} \arrow[l] \\
\text{linear combination } \vec{u} = \sum_i \omega_i \cdot \vec{v}_i \arrow[u] \arrow[r] & \text{descendant } x \arrow[u]
\end{tikzcd}$$

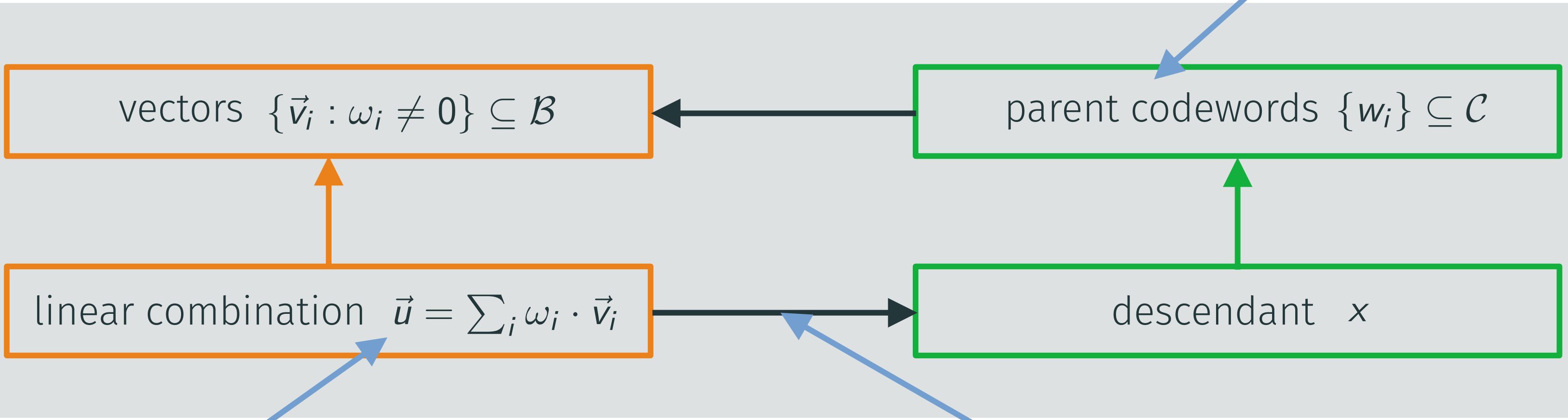
To hide “bad” linear combinations,
we need a hard discrete logarithm ...

Linear-Subspace Tracing: Under the Hood (if time permits...)

Analogy to IPP (and fingerprinting) codes:

- vectors $\longleftrightarrow$ codewords
- linear combinations $\longleftrightarrow$ descendants

IPP codes trace only a *single* parent, so we introduce *fully* IPP codes



To hide “bad” linear combinations, we need a hard discrete logarithm ...

... but then extracting the descendant is quite challenging ...

Conclusion

Conclusion

- definition of linearly-homomorphic group signatures

Conclusion

- definition of linearly-homomorphic group signatures
- combination of signatures created by different users makes derivation and traceability non-trivial

Conclusion

- definition of linearly-homomorphic group signatures
- combination of signatures created by different users makes derivation and traceability non-trivial
- the use of the *subspace-tracing* technique leads to “short” signatures

Conclusion

- definition of linearly-homomorphic group signatures
- combination of signatures created by different users makes derivation and traceability non-trivial
- the use of the *subspace-tracing* technique leads to “short” signatures

Thank you for your attention!

Conclusion

- definition of linearly-homomorphic group signatures
- combination of signatures created by different users makes derivation and traceability non-trivial
- the use of the *subspace-tracing* technique leads to “short” signatures

Thank you for your attention!

https://eprint.iacr.org/2023/138