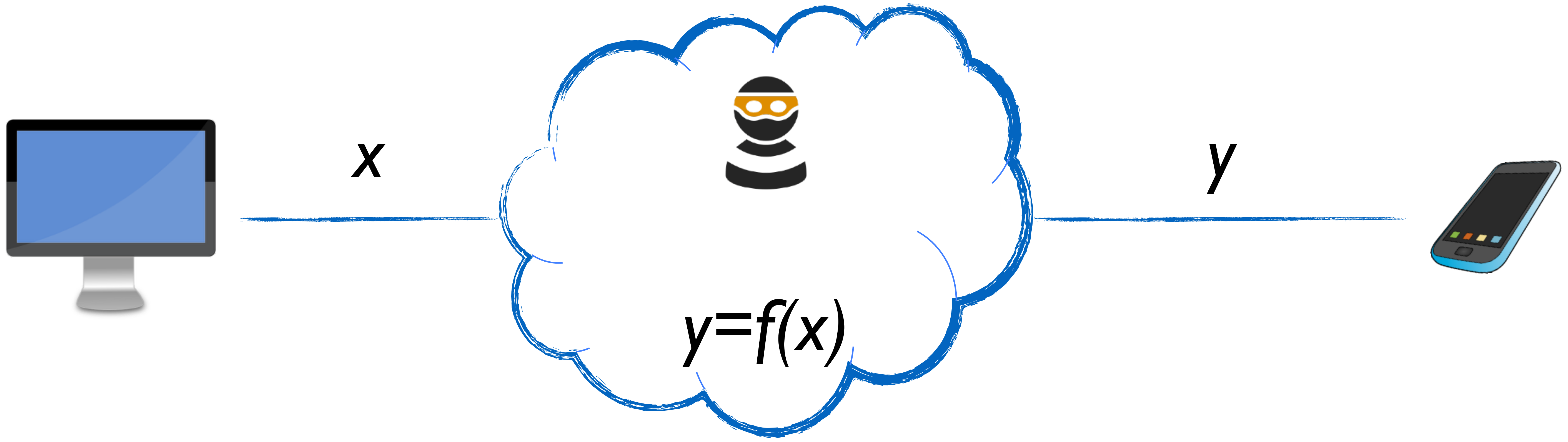
Homomorphic Authentication for Computing Securely on Untrusted Machines

Dario Fiore **IMDEA Software Institute, Spain**

Paris Crypto Day — March 18, 2019

computing on untrusted machines



devices receive information processed on untrusted machines

security concerns

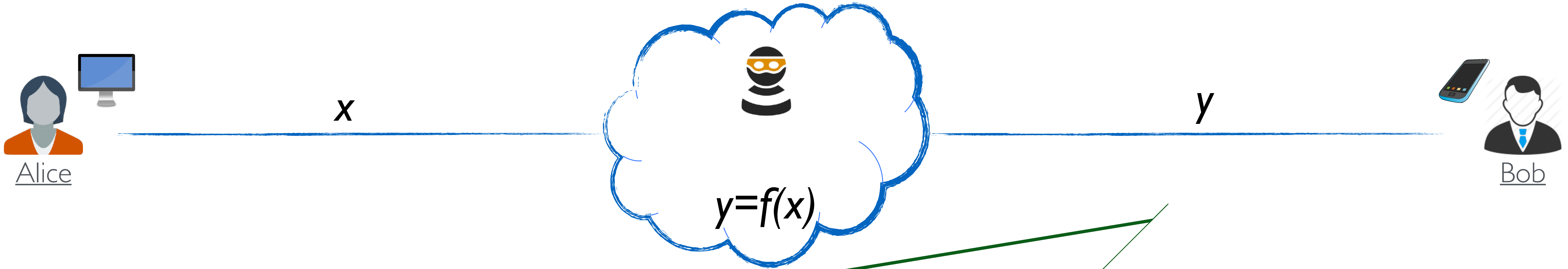
integrity. ensuring that results computed by third parties are correct?

privacy. ensuring that no unauthorized information is leaked to the third parties?

real world

selection of security incidents

main security goals / research problems



computation's integrity. ensuring correctness of computations performed by untrusted machines. *Bob must efficiently establish if y=f(x), given f, x, y*

verifiable computation

computation's authenticity. ensuring correctness of computation and origin of the data used in the computation performed by untrusted machines

Bob must efficiently establish if y=f(x) for an x from Alice, given f, y

homomorphic authentication

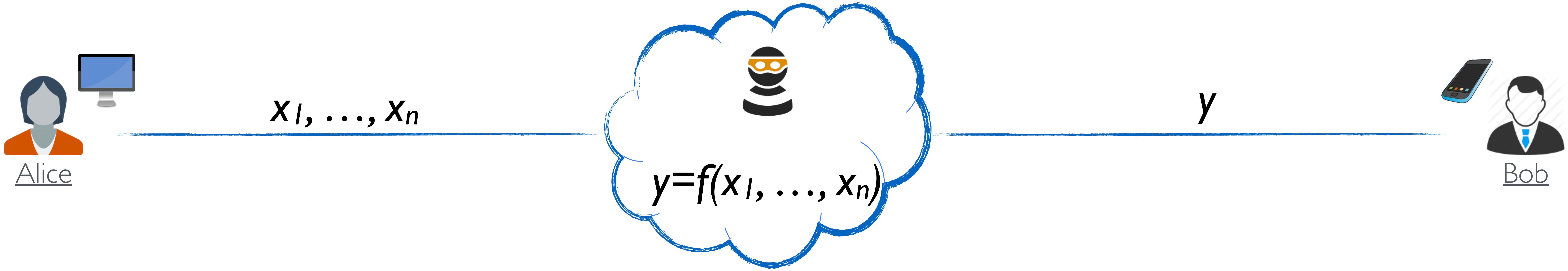
privacy-preserving computation. enabling untrusted machine to compute *f*(*x*) without learning *x* (+ can also ensure integrity/authenticity)

homomorphic/ functional/searchable… encryption

roadmap of this talk

- **computing on untrusted machines**
- **focus on computation authenticity**
  - **homomorphic authentication**
    - concept
    - state of the art
    - a simple realization
  - **computation authenticity for multiple data sources**
  - **conclusions**

computation's authenticity



main desiderata

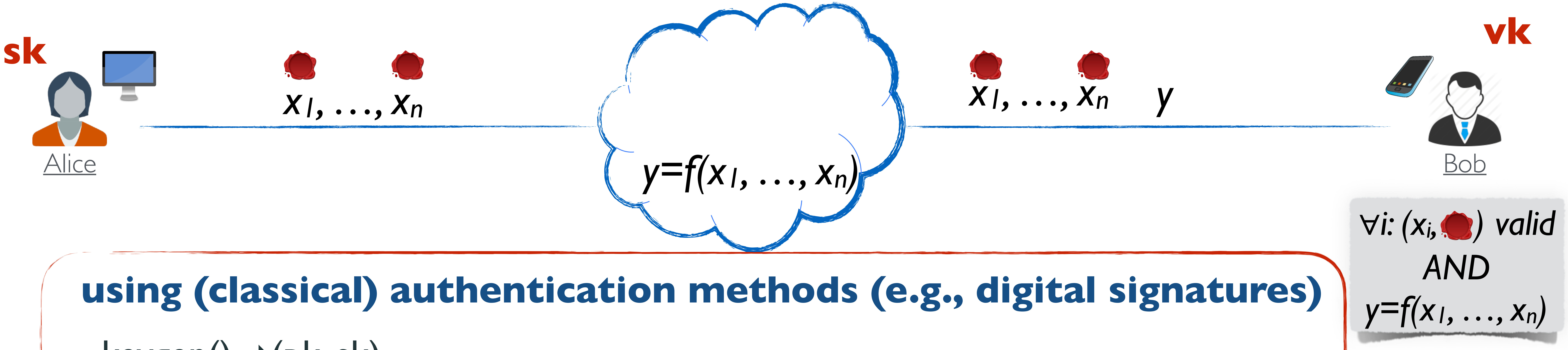
security/authenticity. untrusted machine unable to cheat (i.e., sending $y' \neq f(x_1, \ldots, x_n)$) + Bob must get convinced that data from Alice used to obtain y

efficiency. communication/storage of Bob minimized

challenge. achieving both security and efficiency

how to achieve only efficiency (w/o security)?

a solution with security and without efficiency



using (classical) authentication methods (e.g., digital signatures)

keygen()→(pk, sk)

sign(sk, m)→s

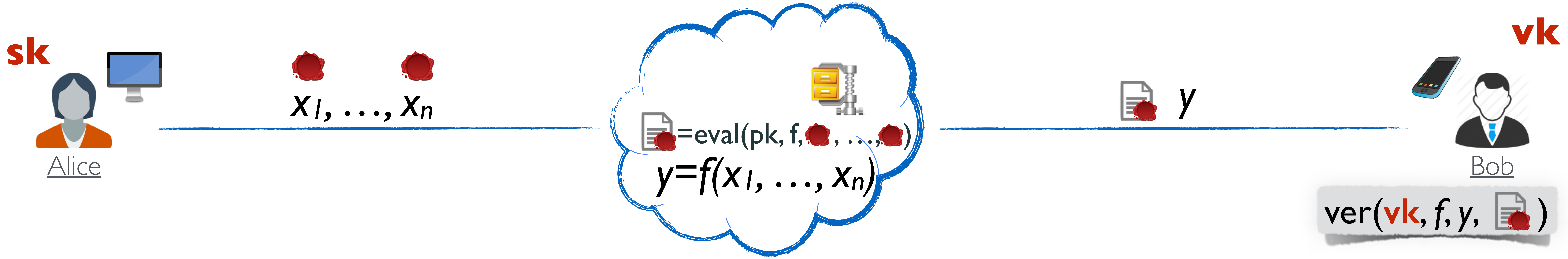
ver(pk, m, s)→{reject, accept}

security guarantee (unforgeability): w/o sk not possible to generate a valid signature

security/authenticity. Cloud unable to cheat (e.g., sending $y'=f'(x_1, \ldots, x_n)$, or $y''=f(x'_1, \ldots, x_n)$)

efficiency. communication/storage of Bob minimized

security & efficiency: homomorphic authentication



security/authenticity. w/o sk only possible to certify correct computations results ⇒ Cloud cannot cheat

efficiency. size of authenticators independent of *n*
⇒ communication/storage of Bob minimized

homomorphic authentication

- concept introduced by [Desdmedt93]
- first formalization by [Johnson-Molnar-Song-Wagner02]
- formal definitions by [Boneh-Feeeman-Katz-Waters09] (network coding application)
- first full fledged formalization [Boneh-Freeman11]

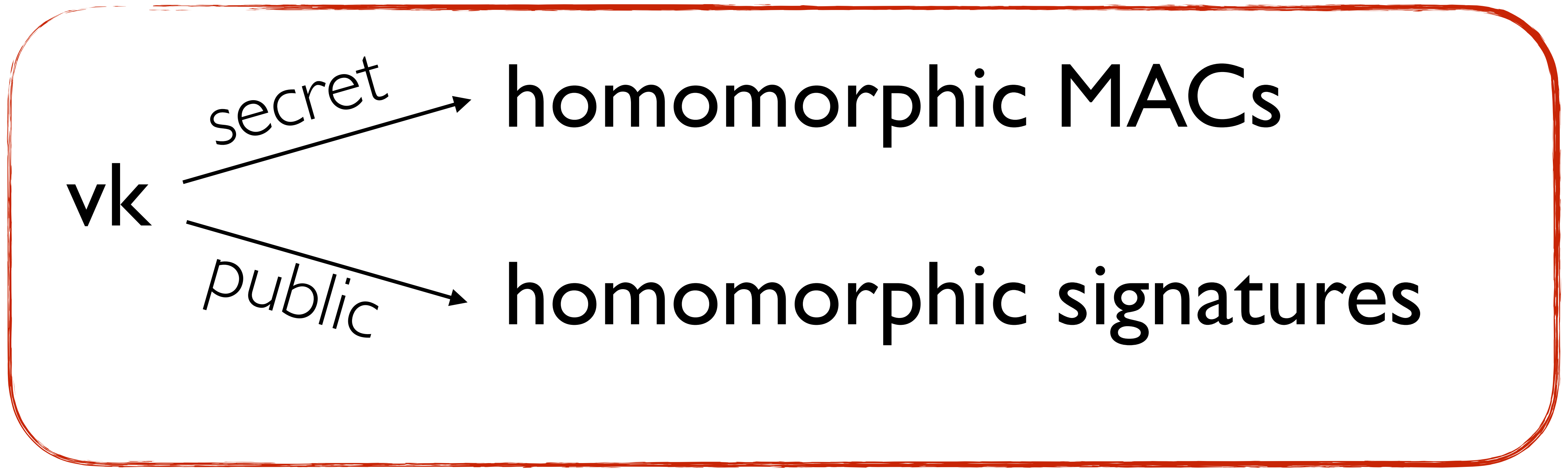
homomorphic authenticators (HA)

keygen(1^k)$\rightarrow$(sk, ek, vk)

auth(sk, **i**, x_i)$\rightarrow\sigma_i$

eval(ek, f, $\sigma_1, \ldots, \sigma_n$)$\rightarrow\sigma$

ver(vk, **f**, y, σ)$\rightarrow${reject, accept}



correctness (basic idea).

{$\sigma_i\leftarrow$**auth**(sk, **i**, x_i)} and $\sigma\leftarrow$**eval**(ek, f, $\sigma_1, \ldots, \sigma_n$),

$\Rightarrow$ **ver**(vk, **f**, f($x_1,\ldots,x_n$), σ)=accept

succinctness. there is a universal polynomial p( ) such that $|\sigma|\leq$p(k, log n)

security. w/o sk one can only create valid authenticators on legitimate outputs

* deliberately omitting some details of the model for simplicity

unforgeability of homomorphic authenticators

adversary wins if

$$y^* \neq f(x_1, \ldots, x_n) \ \text{ AND } \ \textbf{ver}(vk, f, y^*, \sigma^*) = \text{accept}$$

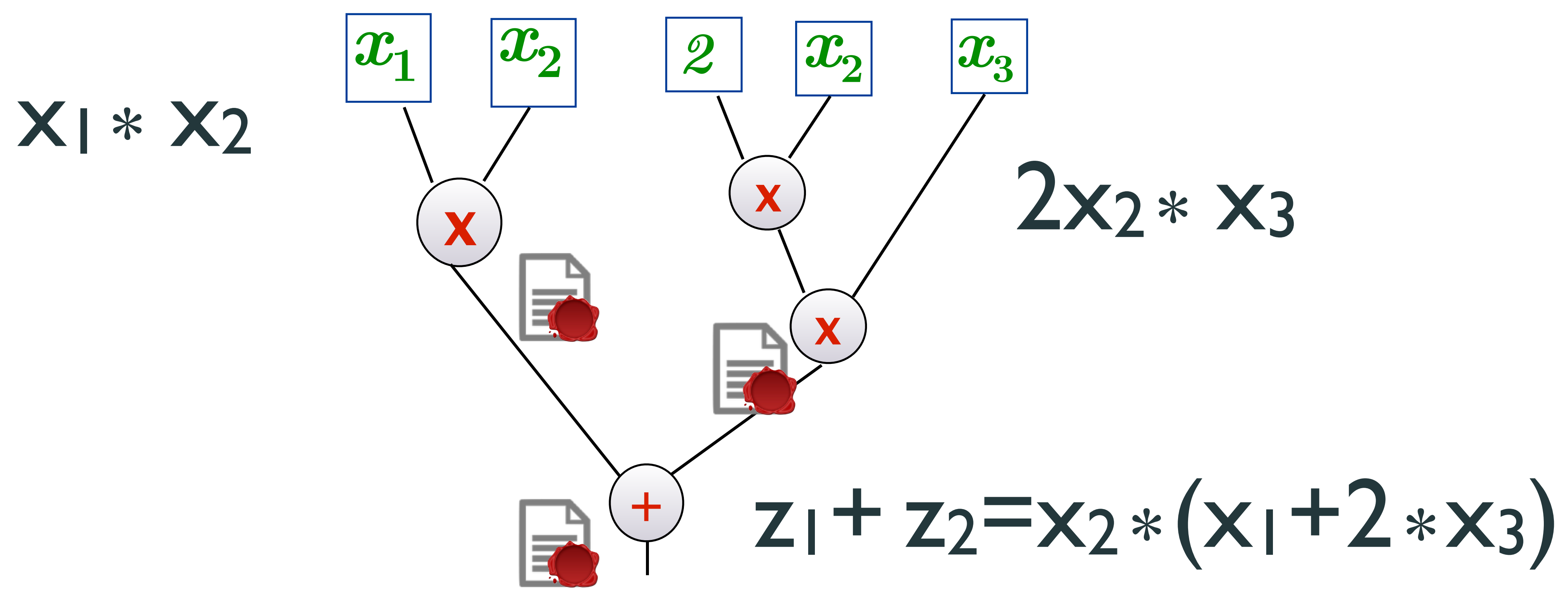
unforgeability. an HA scheme is unforgeable if any PPT adversary wins this game with negligible probability

def. subtleties. how to define forgeries if some i was never queried?

[CFN18] simply say it is a forgery if inputs are missing

additional (interesting) properties of HAs

composability. outputs of **eval** can be fed back to **eval**



useful to parallelize/distribute computation with correctness proofs

context-hiding. authenticators on functions outputs do not reveal information about the inputs

roadmap of this talk

- **computing on untrusted machines**
- **focus on computation authenticity**
  - **homomorphic authentication**
    - concept
    - state of the art
    - a simple realization
  - **computation authenticity for multiple data sources**
  - **conclusions**

HA from the origins to state-of-the-art

the concept of homomorphic authentication

- concept introduced by [Desdmedt93]
- first formalization by [Johnson-Molnar-Song-Wagner02]
- formal definitions by [Boneh-Feeeman-Katz-Waters09] (network coding application), [Boneh-Freeman11] (first full-fledged formalization)

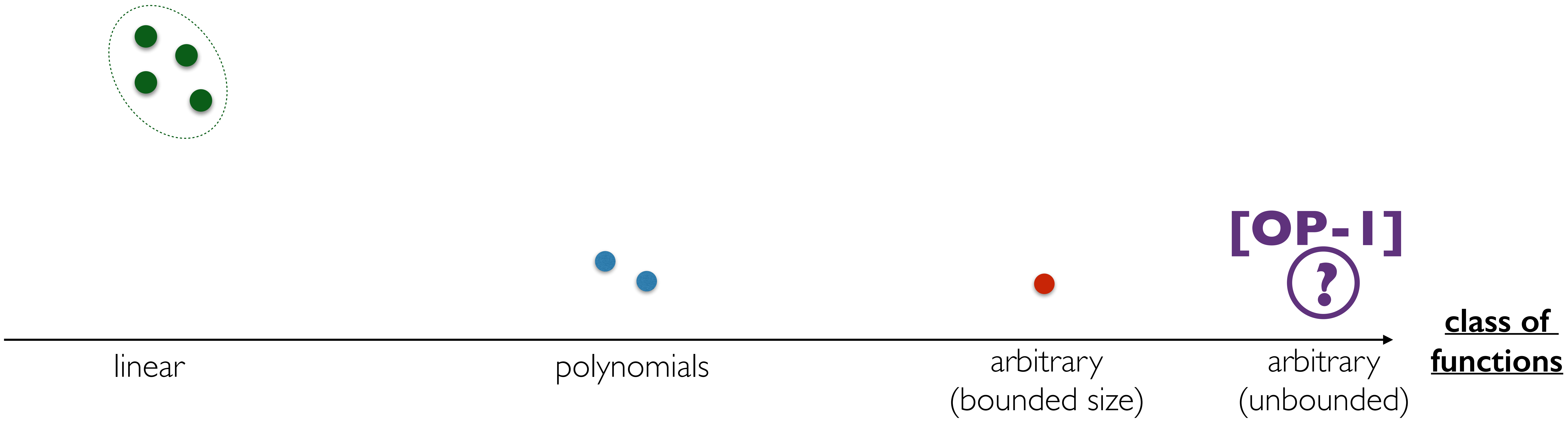
two fundamental research directions

(1) to broaden the class of functionalities that can be computed homomorphically

(2) to obtain efficient instantiations

(I) supported functionality (HS)

HS state of the art



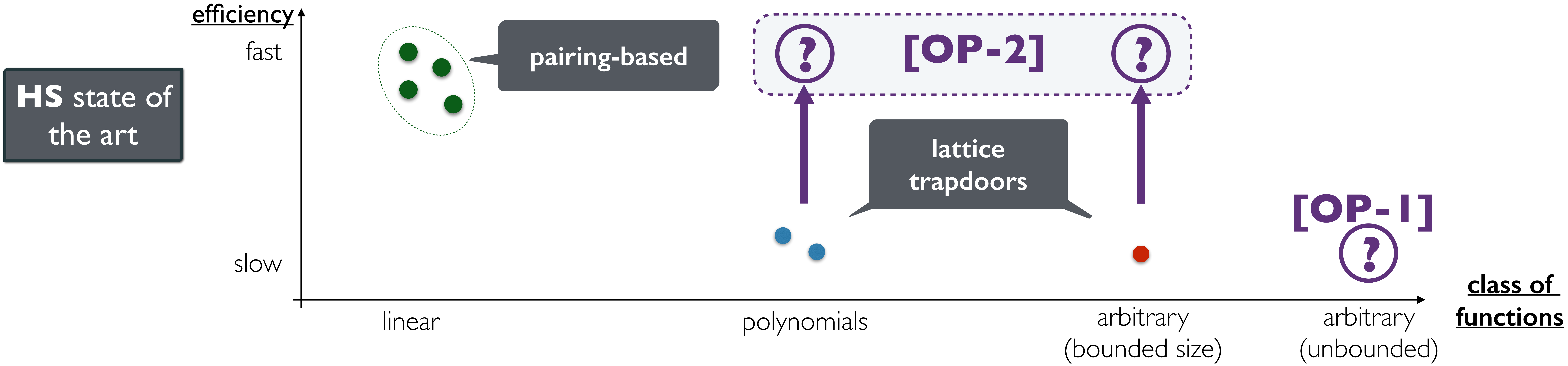
linear functions [Boneh-Freeman-Katz-Waters09, Gennaro-Krawczyk-Rabin10, Catalano-F-Warinschi11, Attrapadung-Libert11, Catalano-F-Warinschi12, Catalano-F-Gennaro-Vamvourellis13, Libert-Peters-Joye-Yung13, Catalano-F-Nizzardo15, ……]

low-degree polynomials [Boneh-Freeman11, Catalano-F-Warinschi14]

arbitrary circuits of bounded depth [Gorbunov-Vaikunthanan-Wichs15]

arbitrary circuits (fully homomorphic) [OP-1] ?

(2) efficiency of HS constructions



linear functions [Boneh-Freeman-Katz-Waters09, Gennaro-Krawczyk-Rabin10, Catalano-F-Warinschi11, Attrapadung-Libert11, Catalano-F-Warinschi12, Catalano-F-Gennaro-Vamvourellis13, Libert-Peters-Joye-Yung13, Catalano-F-Nizzardo15, ……]

low-degree polynomials [Boneh-Freeman11, Catalano-F-Warinschi14]

arbitrary circuits of bounded depth [Gorbunov-Vaikunthanan-Wichs15]

arbitrary circuits (fully homomorphic) [OP-1] ?

fast&expressive HS [OP-2] ?

functionality&efficiency of Hom. MACs constructions

HMACs state of the art

arbitrary circuits [Gennaro-Wichs13] (no verification queries supported)

low-degree arithmetic circuits (NC1) [Catalano-F13, Catalano-F-Nizzardo14]

degree-2 arithmetic circuits [Backes-F-Reischuk13, F-Gennaro-Pastro14]

(new property: efficient verification)

efficient FH-MACs [OP-3] / FH MACs secure w/verification queries [OP-4]

? ?

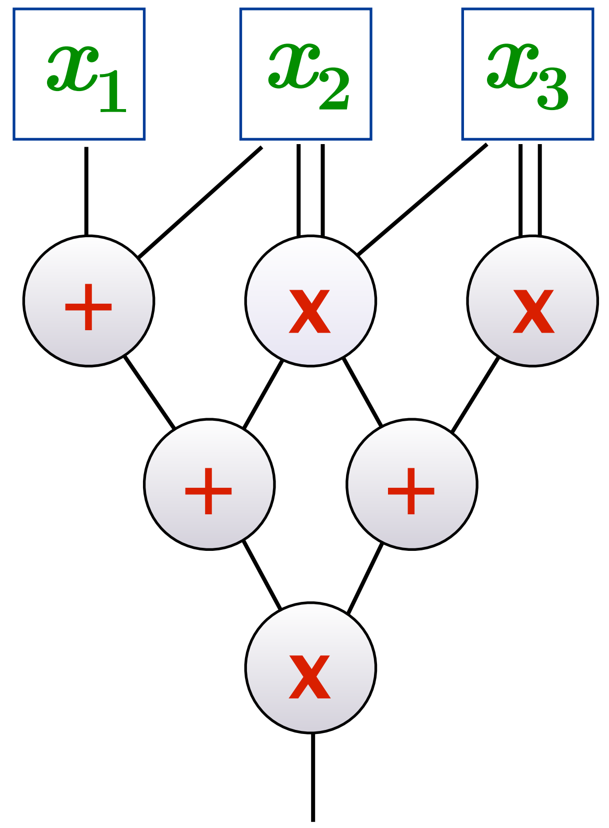
roadmap of this talk

- **computing on untrusted machines**
- **focus on computation authenticity**
- **homomorphic authentication**
  - concept
  - state of the art
  - a simple realization
- **computation authenticity for multiple data sources**
- **conclusions**

a simple and practical homomorphic MAC [CF13]

inputs. values $x_i \in \mathbb{Z}_p$

computations. arithmetic circuits of low degree over



applications.

computations expressible w/boolean circuits of logarithmic depth (NC^1)

arithmetic computations: polynomials, linear algebra, …

CF13 homomorphic MAC

keygen()

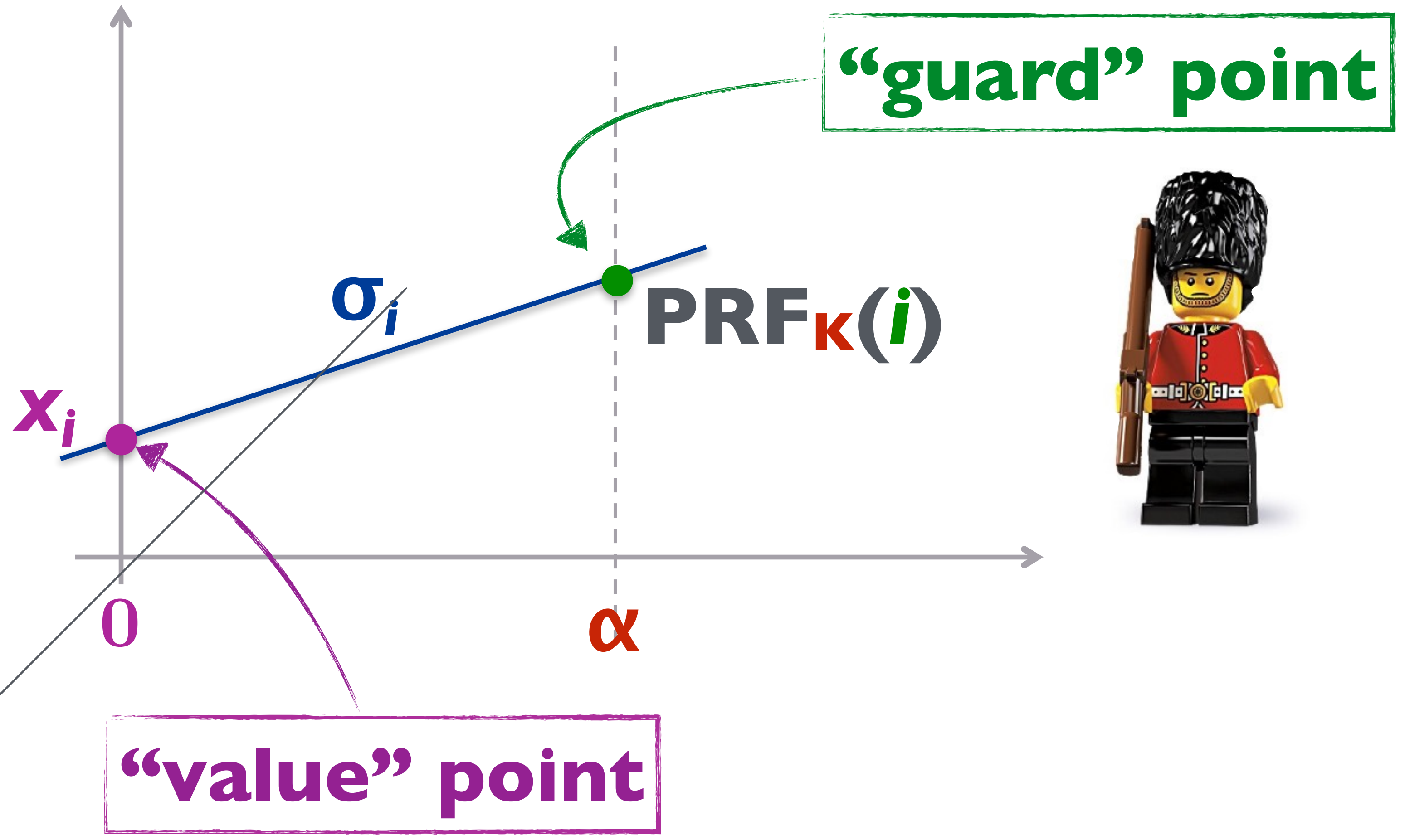
choose the **key K** of a **PRF$_K$**
and a secret line $\alpha \in \mathbb{Z}_p$
sk=(K, α)

auth(sk, i, x$_i$)

Encode **value x_i** (an integer)
with **label/index i**
as a **polynomial σ_i(Z)**
of degree 1 such that:

$\sigma_i(\alpha)$ = **PRF$_K$(i)**

$\sigma_i(0) = x_i$



$\sigma_{i,0} = x_i$, $\sigma_{i,1}$ =(PRF$_K$(i) - x_i)/α

ver(sk, i, x$_i$, σ_i)

Check the "guard" point
i.e., recompute **PRF$_K$(i)** and
evaluate σ_i on **0** and α

the CF13 homomorphic MAC

eval(f, $\sigma_1, \ldots, \sigma_k$)

point-wise execution of arithmetic operations

$$\sigma^*(Z) = f(\sigma_1(Z), \ldots, \sigma_k(Z))$$

addition: addition of coefficients

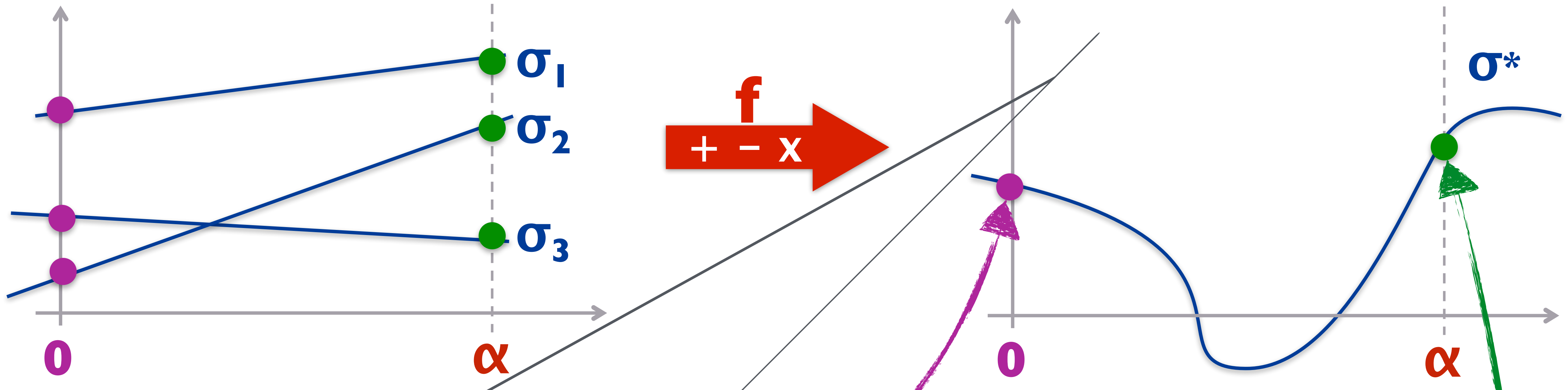
multiplication: convolution of polynomials

ver(sk, f, y, σ^*)

Check

$\sigma^*(\alpha) = f(PRF_K(1), \ldots, PRF_K(k))$

$\sigma^*(0) = y$



correctness:

result $\sigma^*(0) = f(\sigma_1(0), \ldots, \sigma_k(0))$
$= f(x_1, \ldots, x_k)$

"guard" $\sigma^*(\alpha) = f(\sigma_1(\alpha), \ldots, \sigma_k(\alpha))$
$= f(PRF_K(1), \ldots, PRF_K(k))$

unforgeability.

intuition: unpredictability of the guard point

a bit more precisely:

PRF security + Schwartz-Zippel

succinctness. $|\sigma^*| = O(\deg(f))$

or $|\sigma^*| = O(1)$ under $\deg(f)$-DH assumption

HAs with efficient verification

CF13 verification requires recomputing f

how to verify efficiently?

[BFR13] introduced the model and a first realization

basic idea.

$\mathsf{ver}(vk, \mathbf{f}, y, \sigma)$

$\mathsf{ver}_{off}(vk, \mathbf{f}) \rightarrow vk_{\mathbf{f}}$

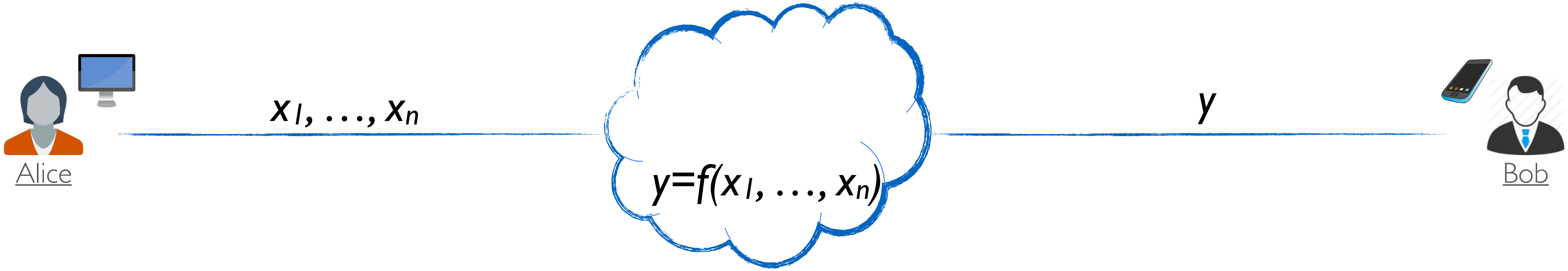
$\mathsf{ver}_{on}(vk_{\mathbf{f}}, y, \sigma) \rightarrow \{\text{reject}, \text{accept}\}$

this is by now a desired verification model (also in homomorphic signatures)

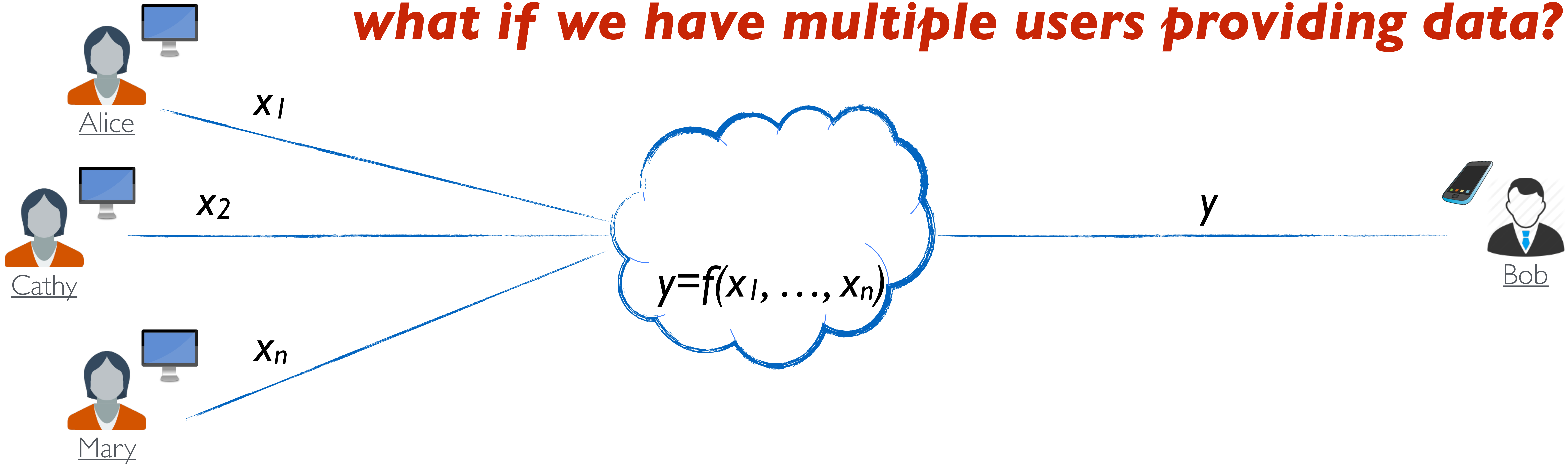
roadmap of this talk

- **computing on untrusted machines**
- **focus on computation authenticity**
- **homomorphic authentication**
  - concept
  - state of the art
  - a simple realization
- **computation authenticity for multiple data sources**
- **conclusions**

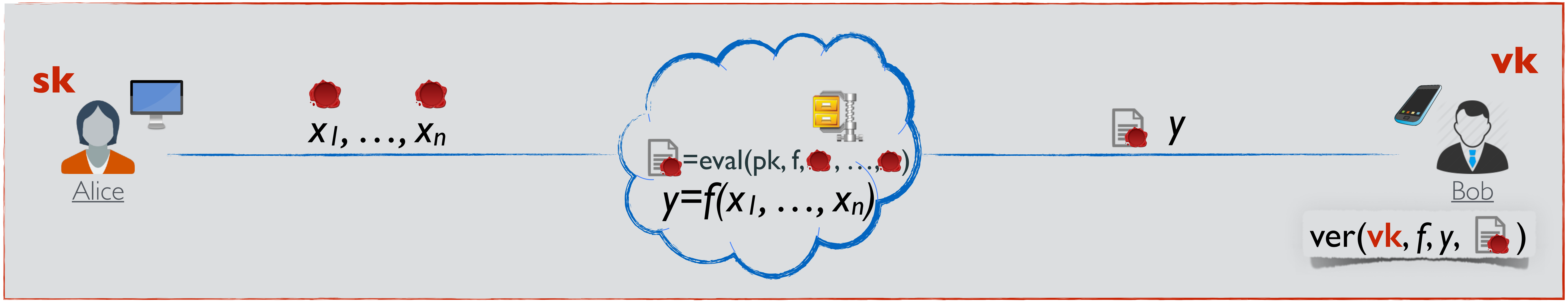
computation authenticity for multiple users



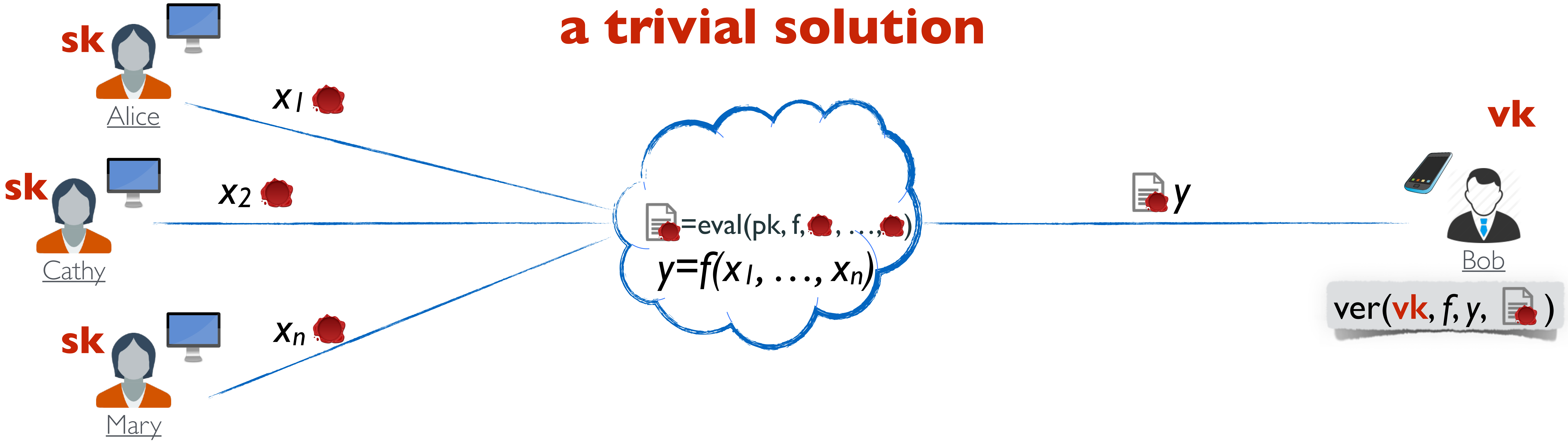
what if we have multiple users providing data?



using (single-user) HAs



a trivial solution



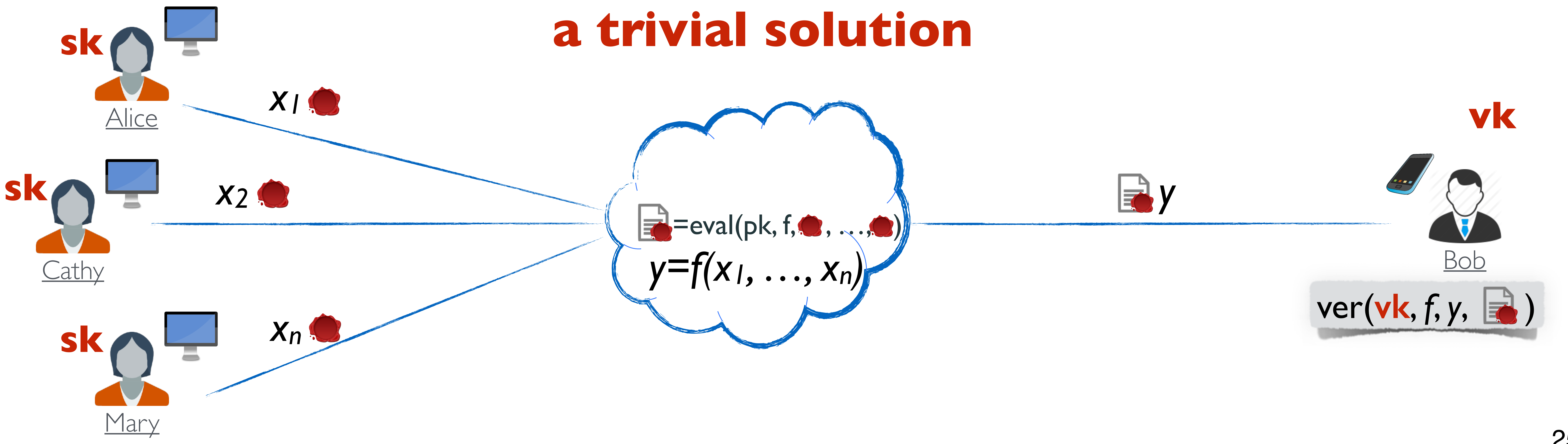
using (single-user) HAs

main issues.

establishing origin. not really… all users look the same

fault tolerance. if one users is compromised all system is compromised!

a trivial solution



multi-key homomorphic authenticators

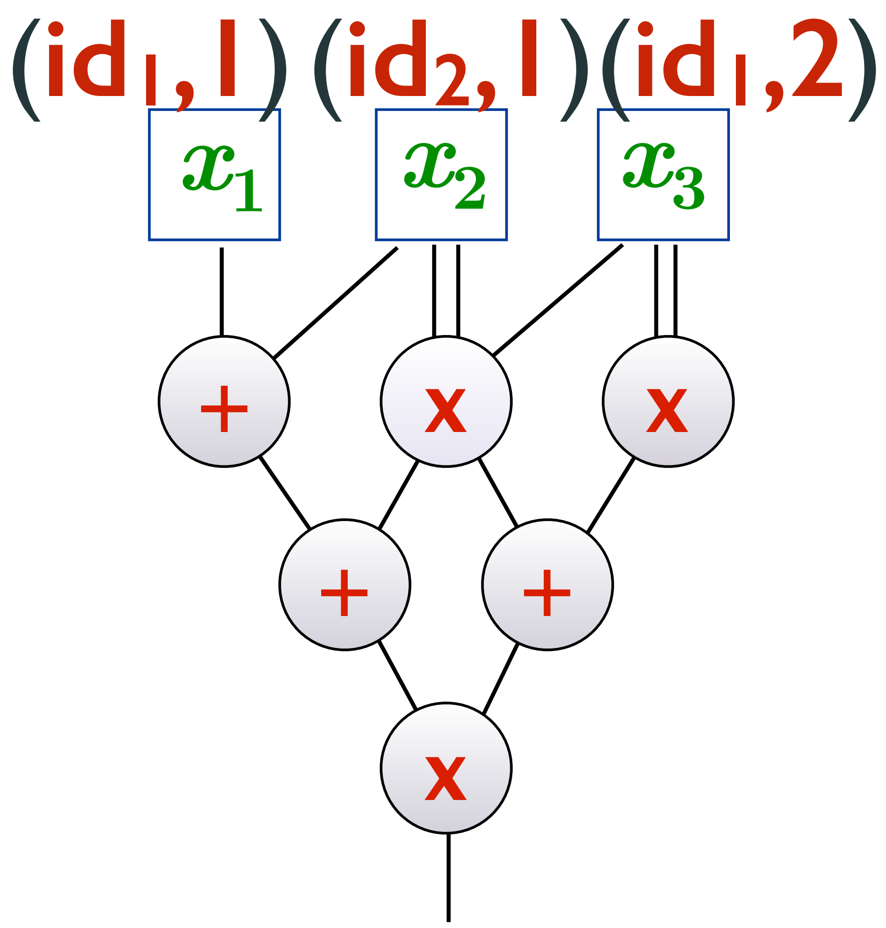
[F-Mitrokotsa-Nizzardo-Pagnin16]

key property. can certify computations on data authenticated with different secret keys

unforgeability. untrusted machine cannot cheat (unless it learns some sk_i involved in the computation)

succinctness. size of σ independent of #inputs (but may depend on #users)

multi-key homomorphic authenticators (MK-HA)



setup$(1^k) \rightarrow pp$

keygen$(pp) \rightarrow (sk_{id}, ek_{id}, vk_{id})$

auth$(sk_{id}, (\mathbf{id}, \mathbf{i}), x) \rightarrow \sigma_{id,i}$

eval$(f, \{\sigma_i, EKS_i\}_{i=1..n}) \rightarrow \sigma$ // each $EKS_i = \{ek_{id}\}$

ver$(\mathbf{f}, \{vk_{id}\}, y, \sigma) \rightarrow \{\text{reject, accept}\}$

correctness (basic idea).

$$\{\sigma_j \leftarrow \textbf{auth}(sk_{idj}, (\mathbf{id_j}, \mathbf{i_j}), x_j)\} \text{ and } \sigma \leftarrow \textbf{eval}(f, \{\sigma_j, \{ek_{idj}\}\}_{j=1..n}),$$

$$\Rightarrow \textbf{ver}(f, \{vk_{id}\}, f(x_1, \ldots, x_n), \sigma) = \text{accept}$$

succinctness. there is a universal polynomial p(k) such that $|\sigma| \le p(k, n, \log t)$

security. w/o sk of users involved in a computation, one can only create valid authenticators on legitimate outputs

a look at multi-key HAs state of the art

	[F-Mitrokotsa-Nizzardo-Pagnin16]		[Lai et al. 18]
	MK-HS	**MK-HMAC**	**MK-HS*** *stronger security
functions	arbitrary circuits of bounded depth	arithmetic circuits of "low degree"	arbitrary circuits of bounded depth
assumptions	SIS	PRF (OWFs)	SNARKs
succinctness (n=#users, d=deg(f))	O(n)	$O(n^d)$ or $O(d^n)$	O(1)

multi-key HA w/better succinctness from std assumptions? [OP-5] ?

FMNP16 multi-key homomorphic MAC

keygen() at user j

choose the **key K_j** of a **PRF_{Kj}**

and a secret line $\alpha_j \in \mathbb{Z}_p$

$sk_j=(K_j, \alpha_j)$

auth(sk_j, i, x_i)

Encode **value x_i** (an integer)

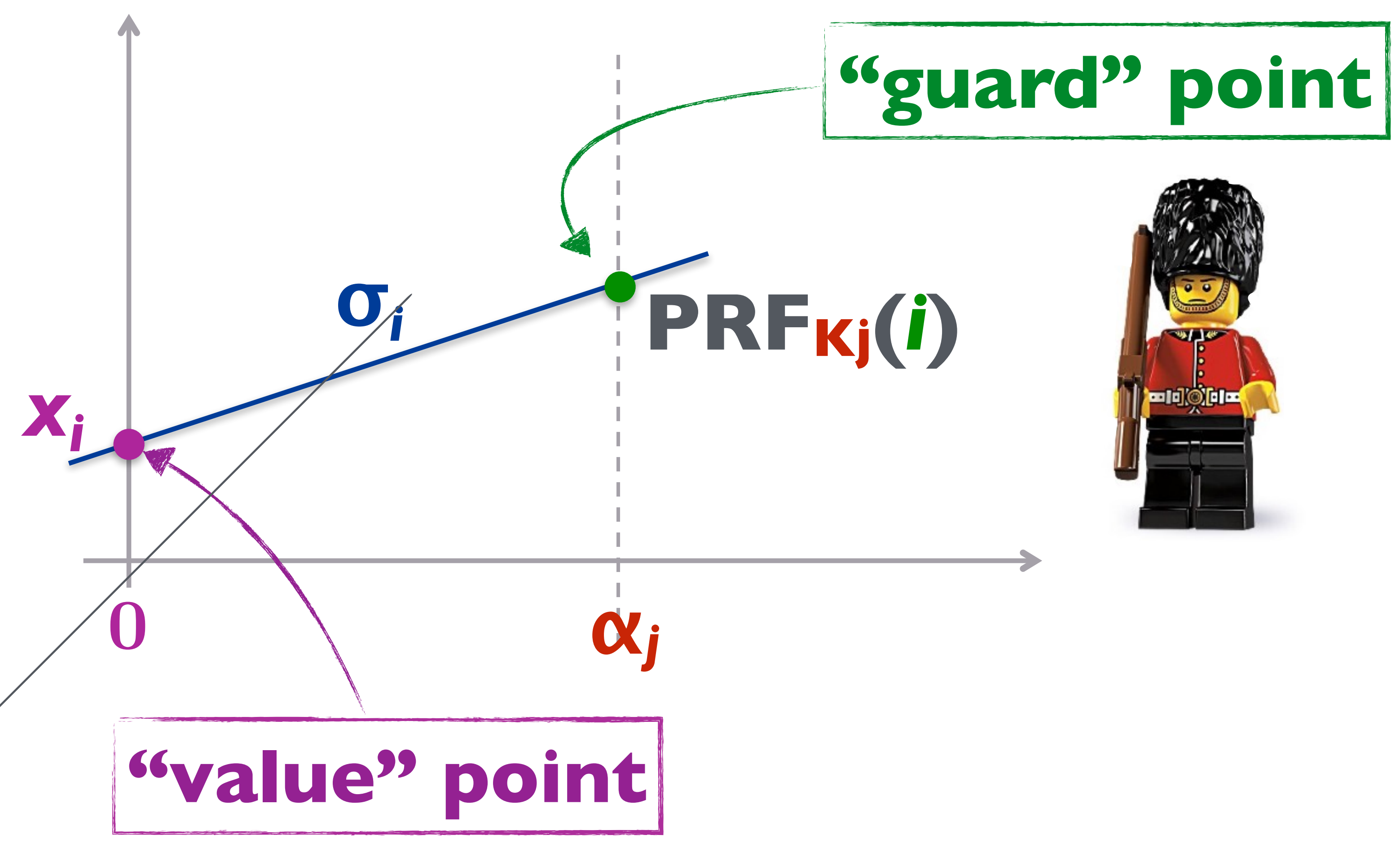
with **label/index *i***

as a **polynomial $\sigma_i(Z_j)$**

of degree 1 such that:

$\sigma_i(\alpha_j) = PRF_{Kj}(i)$

$\sigma_i(0) = x_i$



$\sigma_{i,0} = x_i$, $\sigma_{i,1} = (PRF_{Kj}(i) - x_i)/\alpha_j$

ver(sk_j, i, x_i, σ_i)

Check the "guard" point

i.e., recompute $PRF_{Kj}(i)$ and

evaluate σ_i on **0** and α_j

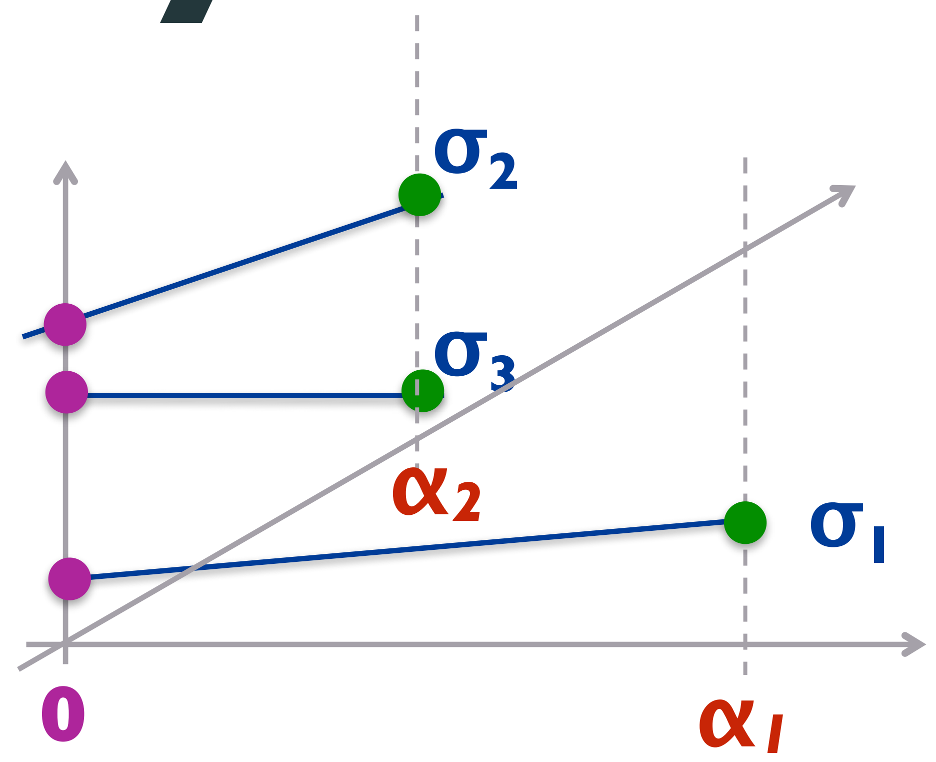
FMNP16 multi-key homomorphic MAC

eval($f, \sigma_1, \ldots, \sigma_k$)

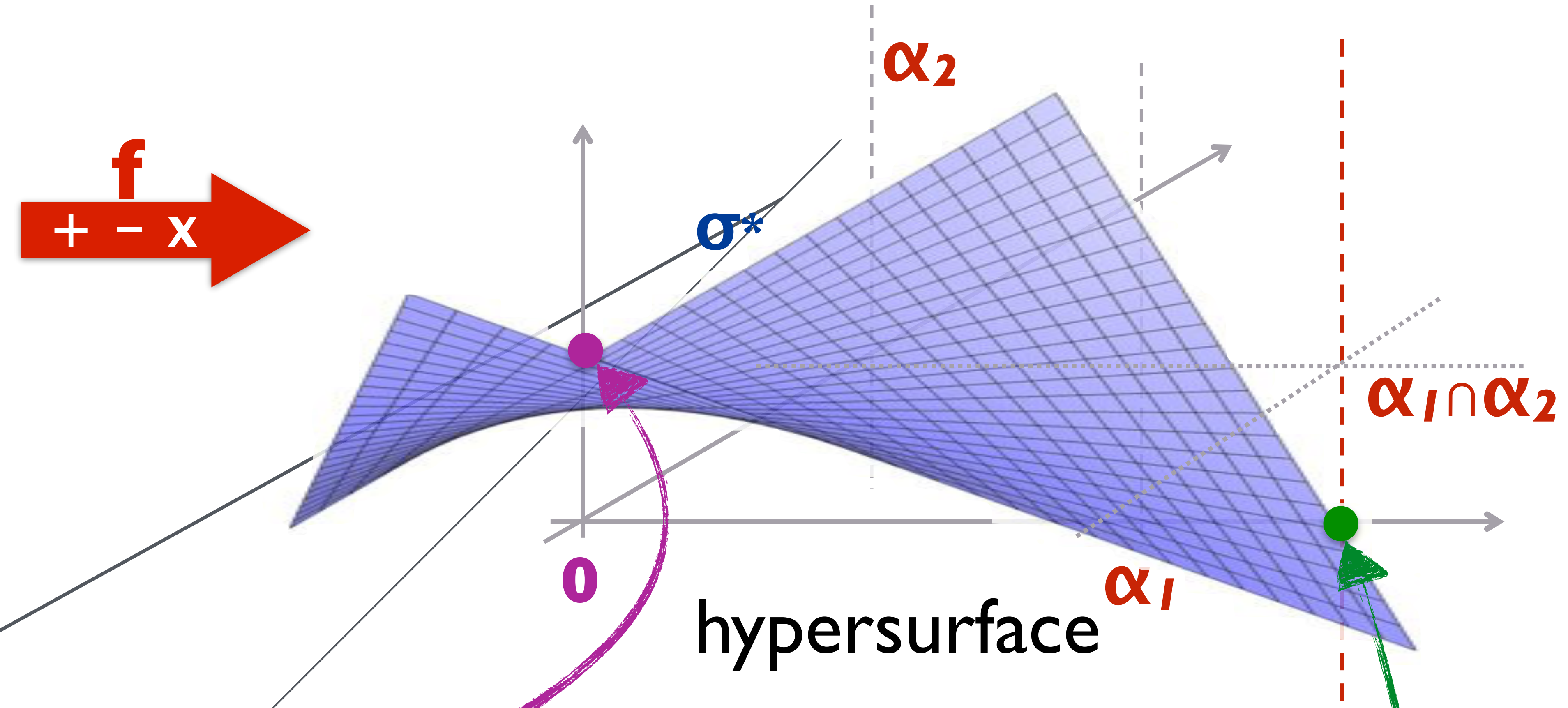
multivariate polynomial evaluation

$$\sigma^*(Z) = f(\sigma_1(Z), \ldots, \sigma_t(Z))$$

$Z = Z_1, \ldots, Z_n$



idea: one plane per user



hypersurface

correctness:

result $\sigma^*(0, \ldots, 0) = f(\sigma_1(0), \ldots, \sigma_t(0))$

$= f(x_1, \ldots, x_t)$

"guard" $\sigma^*(\alpha_1, \ldots, \alpha_n) = f(\sigma_1(\alpha_1), \ldots, \sigma_t(\alpha_t))$

$= f(\mathrm{PRF}_{K1}(1), \ldots, \mathrm{PRF}_{Kt}(t))$

ver(sk, f, y, σ^*)

Check

$\sigma^*(\alpha_1, \ldots, \alpha_n) = f(\mathrm{PRF}_{K1}(1), \ldots, \mathrm{PRF}_{Kt}(t))$

$\sigma^*(0, \ldots, 0) = y$

unforgeability.

intuition: unpredictability of the guard point

(more precisely: PRF + Schwartz-Zippel)

succinctness. $|\sigma^*| = \binom{n+d}{d} = O(n^d)$ or $O(d^n)$

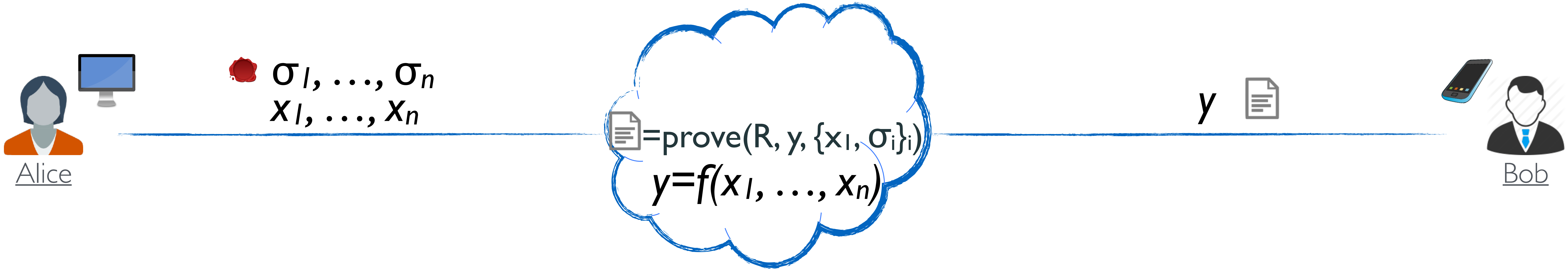
$d = \deg(f)$

roadmap of this talk

- **computing on untrusted machines**
- **focus on computation authenticity**
- **homomorphic authentication**
  - concept
  - state of the art
  - a simple realization
- **computation authenticity for multiple data sources**
- **conclusions**

Alternative Approaches…

computation authenticity via SNARKs



a folklore idea: using SNARKs + digital signatures

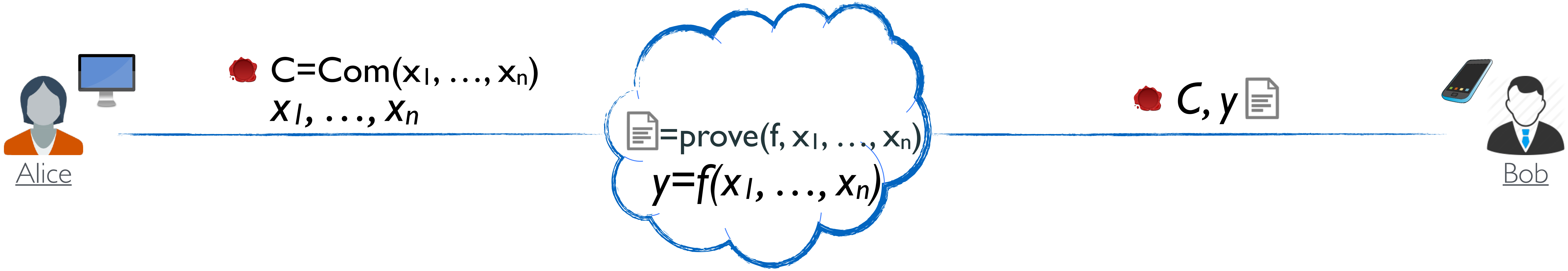
proves that $R(y, \{x_i,\sigma_i\})=1$ iff $y=f(x)$ AND $\forall i$ σ_i is a valid signature on (i, x_i)

SNARK succinctness => HS succinctness

knowledge-soundness + unforgeability => HS unforgeability

…but proving security raises very subtle problems related to extractability

computation authenticity via CP-SNARKs



using commit-and-prove SNARKs + digital signatures

can create proof that $y=f(x)$ w.r.t. $C=Com(x)$ + add signature on commitment C

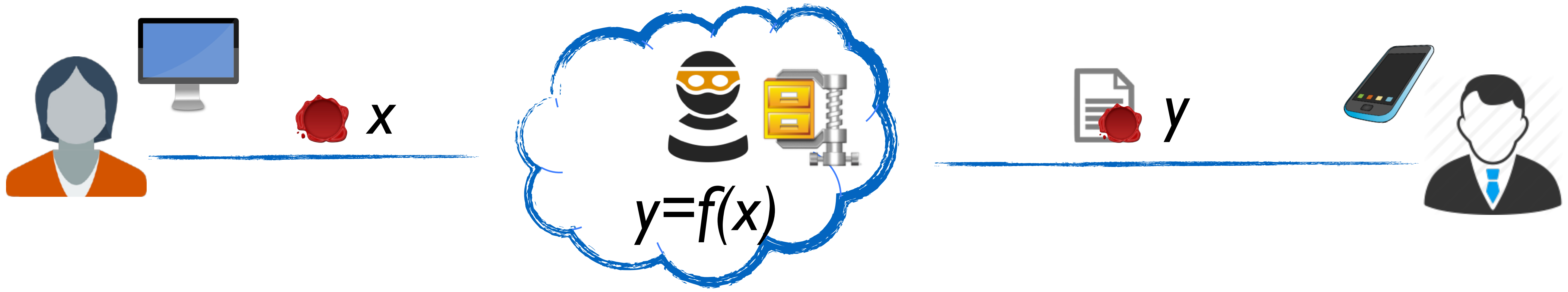
Bob verifies that (C,) is valid signature and that (C, y,) valid proof

“Standard” HA constructions vs. alternative approaches

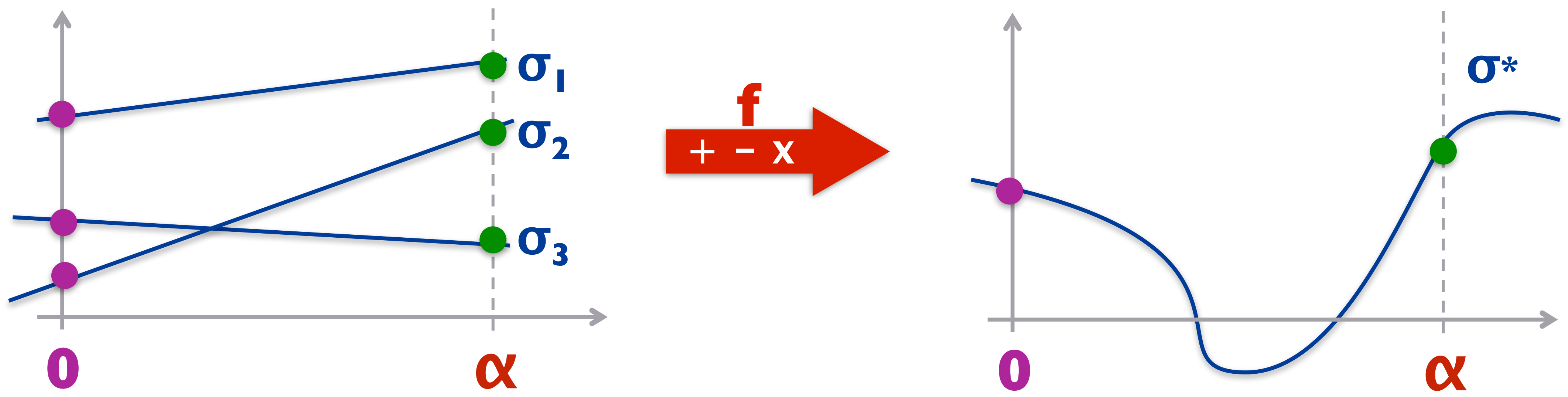
	HA	SNARKs + Signatures	CP-SNARKs + Signatures
efficiency (concrete)	good for linear/ quadratic functions		
assumptions	standard	(oracle) knowledge-type	knowledge-type
public parameters	O(1) (ROM) O(#inputs) (std model)	O(1) ROM O(|f|) **	O(1) ROM O(|f|) **
composition	yes	no*	no*
streaming source	yes	no	yes

conclusions

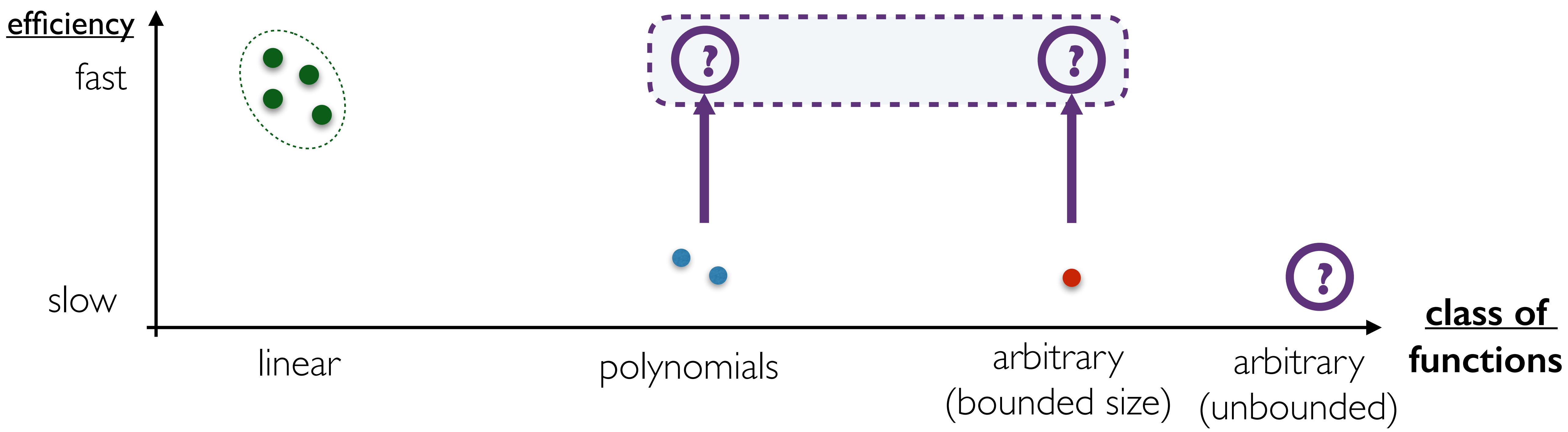
computing securely on untrusted machines with homomorphic authentication



simple homomorphic MACs from OWFs



state of the art



open problems

[OP-1] fully homomorphic signatures

[OP-2] fast&expressive HS

[OP-3] efficient fully homomorphic MACs

[OP-4] fully homomorphic MACs w/ver. queries

[OP-5] fully-succinct multi-key HA

my exciting journey on homomorphic authentication

— Thank you for your attention!

thanks and credit to all my collaborators too!

M. Backes, M. Barbosa, D. Catalano, R. Gennaro, K. Mitrokotsa, L. Nizzardo, E. Pagnin,
V. Pastro, R. Reischuk, B. Warinschi