



Multi-Key Fully-Homomorphic Aggregate MAC for Arithmetic Circuits

Suvasree Biswas and Arkady Yerukhimovich

George Washington University

Outline



1. Motivating Application
2. Tools Used
3. Why Are These Tools Not Enough ?
4. Syntax
5. Security Definition
6. Construction
7. Summary

Motivating Application: Sensor Network



\mathcal{D}_1



\mathcal{D}_2



\mathcal{D}_3

Sensor Mice



Cloud Tom

f_1

f_2

f_3



Scientist Jerry

Motivating Application: Sensor Network



\mathcal{D}_1



\mathcal{D}_2



\mathcal{D}_3

Sensor Mice



Cloud Tom

f_1

f_2

f_3



Scientist Jerry

$$\sum_i f_i(\mathcal{D}_i)$$

Motivating Application: Sensor Network



Sensor Mice



Cloud Tom

$f_1 \mathcal{D}_1$

$f_2 \mathcal{D}_2$

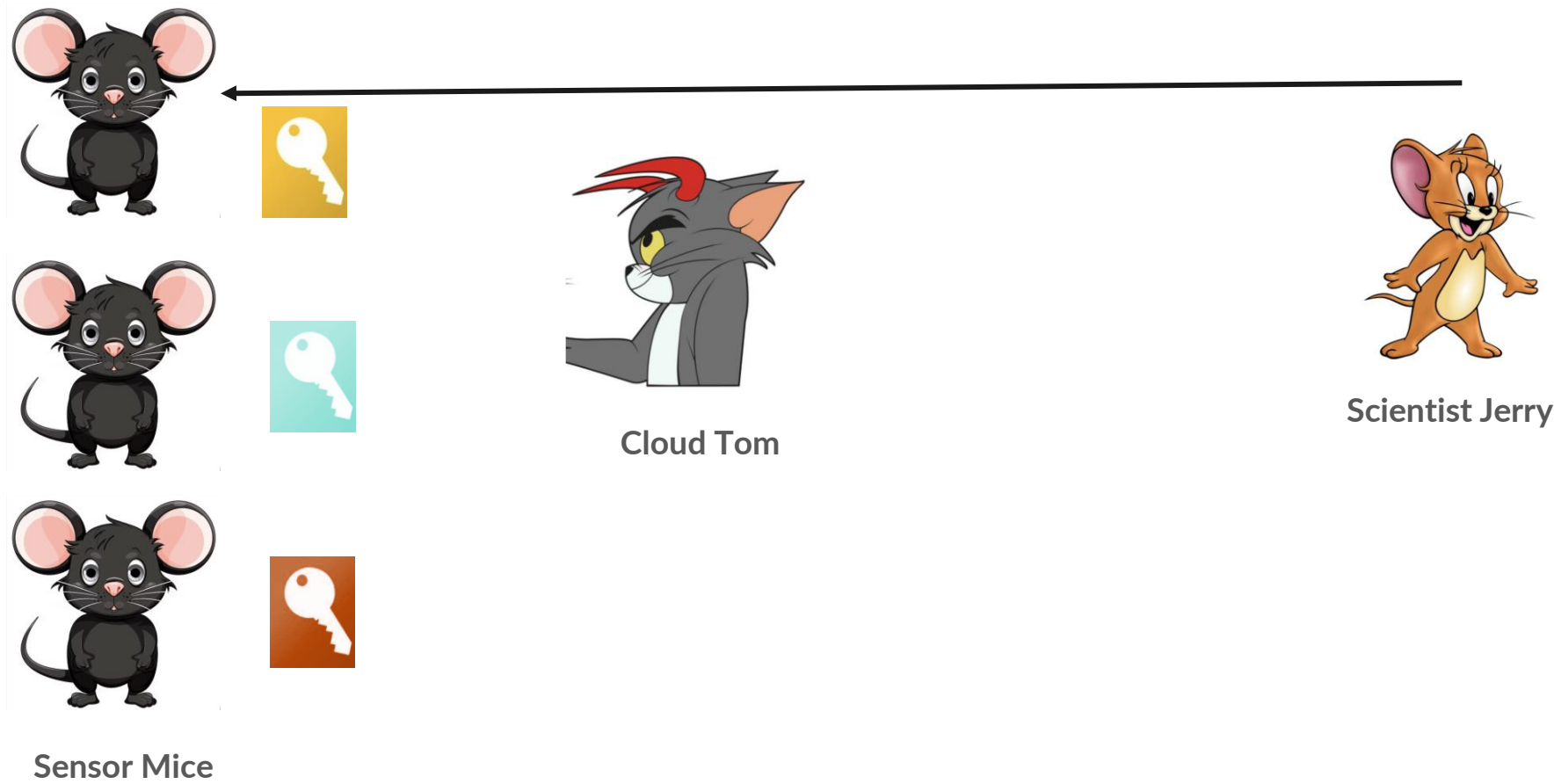
$f_3 \mathcal{D}_3$



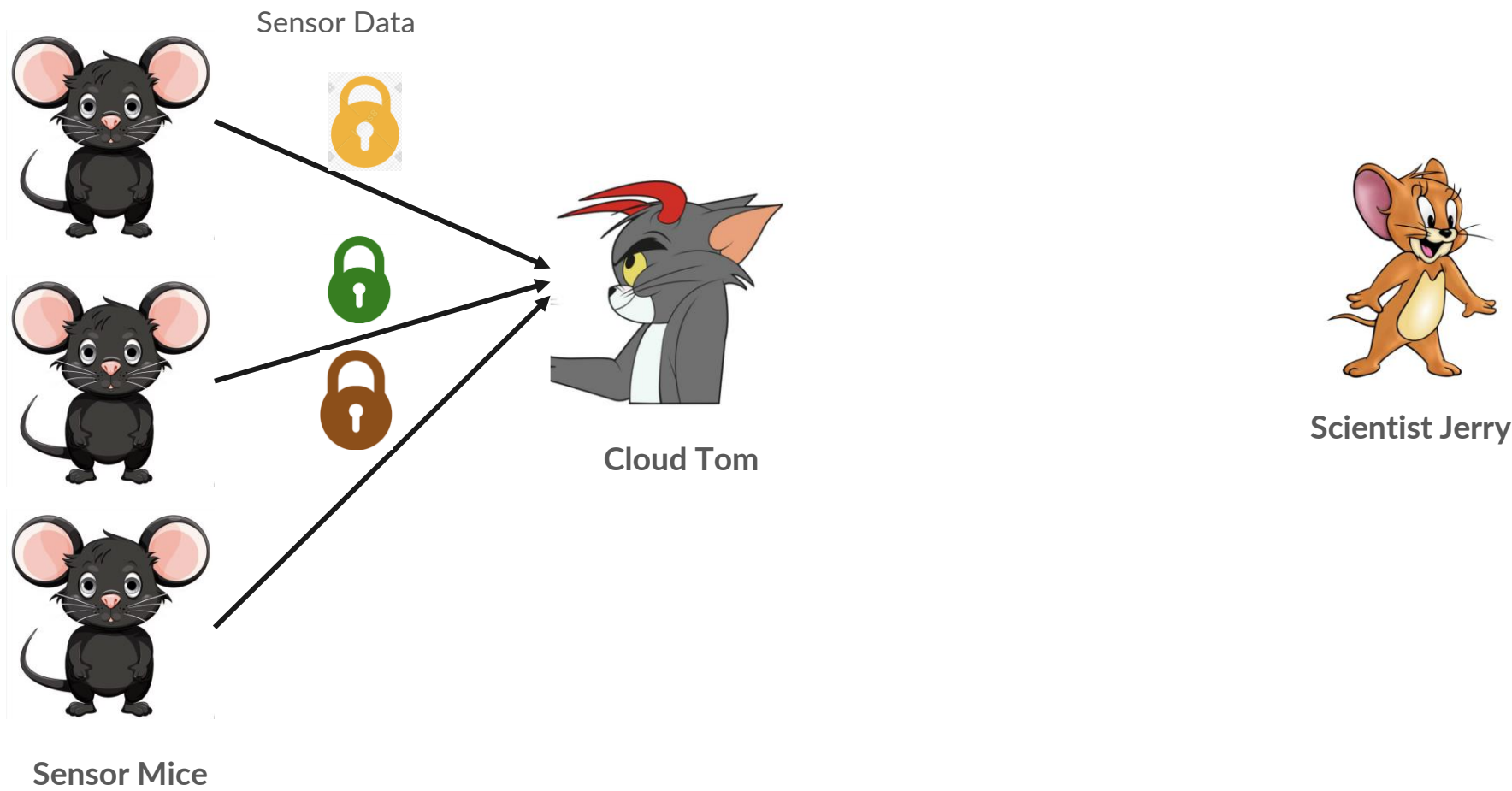
Scientist Jerry

$$\Sigma_i f_i(\mathcal{D}_i)$$

Motivating Application: Sensor Network [Gennaro12](#)



Motivating Application: Sensor Network [Gennaro12](#)



Motivating Application: Sensor Network [Gennaro12](#)



Sensor Mice



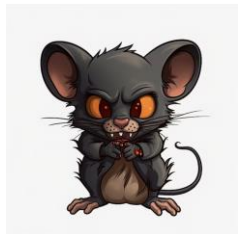
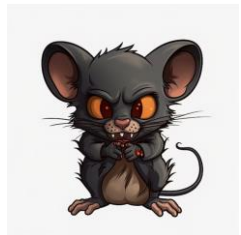
Cloud Tom

Result of computation
→
Proof of correctness of
computation



Scientist Jerry

Motivating Application: Sensor Network [Gennaro12](#)



Sensor Mice



Cloud Tom



Scientist Jerry

Efficiency: proof size independent of function size and number of clients.

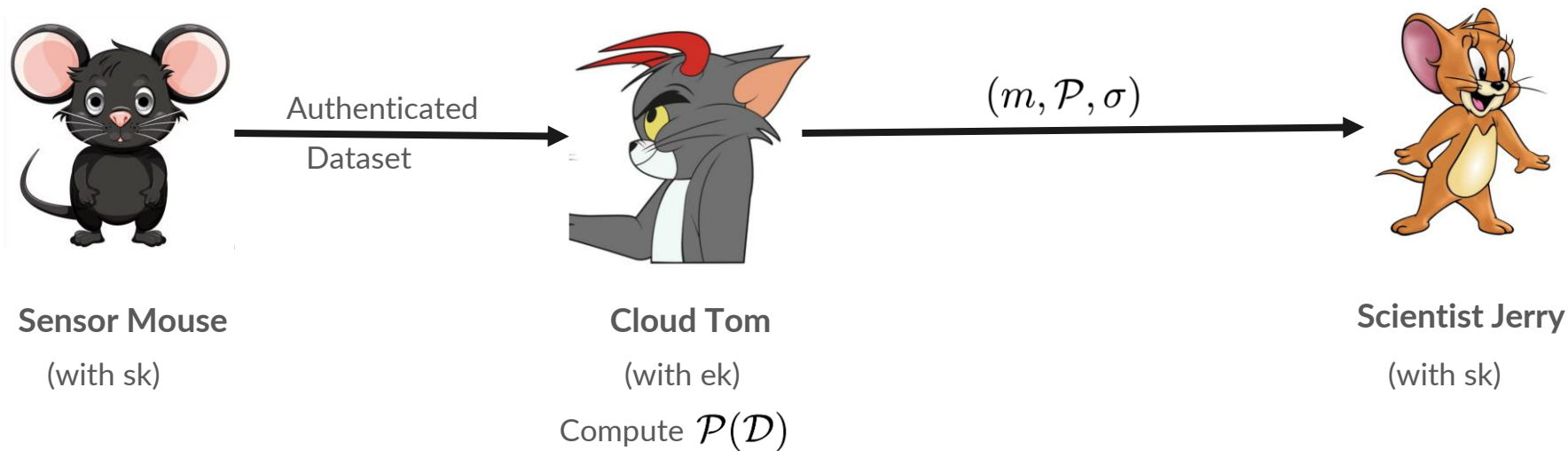
Security: Tom cannot corrupt computation on honest clients' inputs, even if he colluded with some of the clients.

Outline



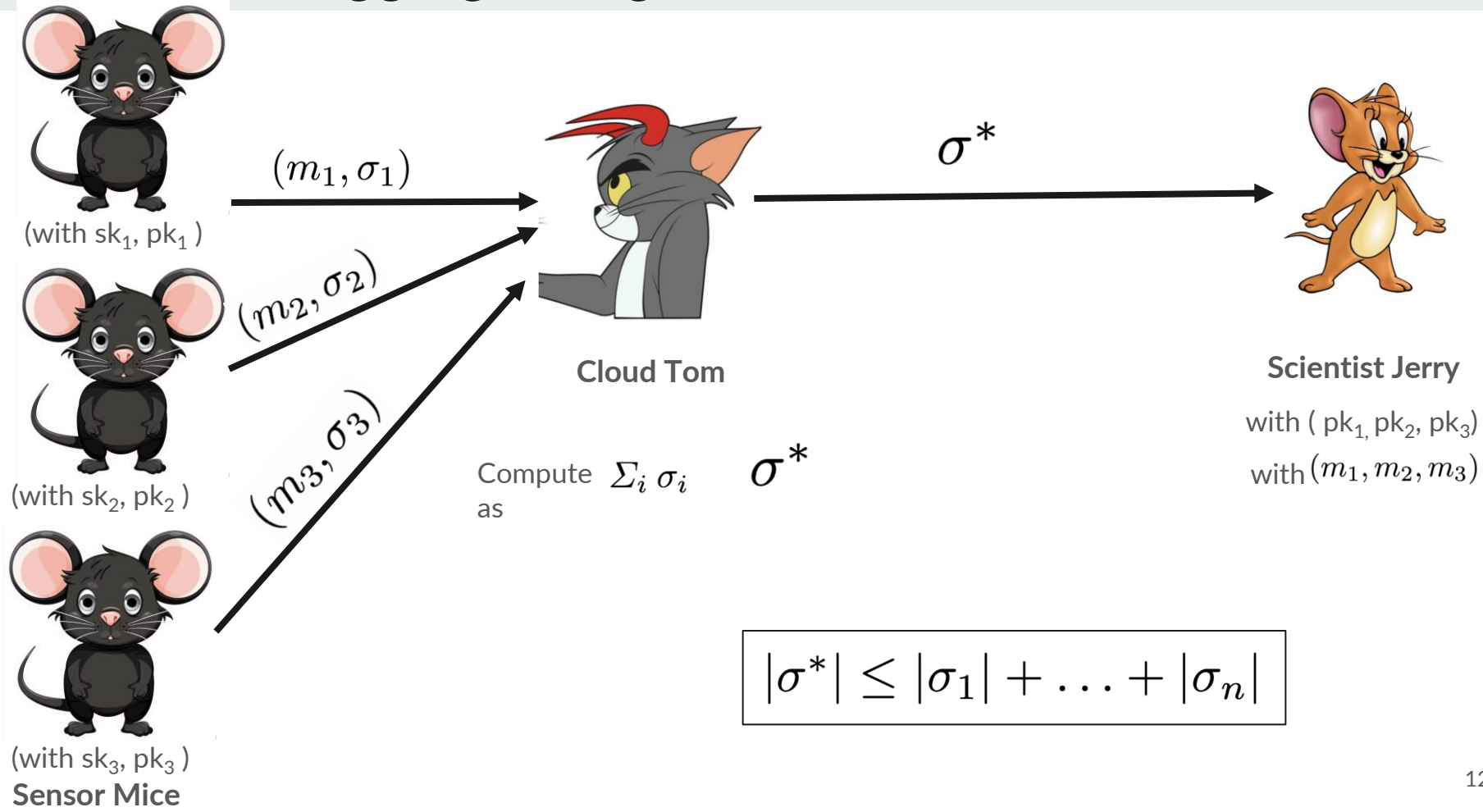
1. Motivating Application
2. Tools Used
3. Why Are These Tools Not Enough ?
4. Syntax
5. Security Definition
6. Construction
7. Summary

First Tool: Homomorphic Authenticators [Gennaro12](#)



- Tag size is independent of function depth and size
- Jerry's Verification is independent of Mouse's dataset

Second Tool : Aggregate Signatures [Boneh13](#)



Outline



1. Motivating Application
2. Tools Used
3. Why Are These Tools Not Enough ?
4. Syntax
5. Security Definition
6. Construction
7. Summary

From Homomorphic Authenticators:

Auth



Sensor Mouse

Eval



Cloud Tom

Keygen

Ver



Scientist Jerry

- Tag size is independent of function depth and size
- Verification is independent of the dataset

From Aggregate Signatures:

Auth



Sensor Mice

Aggregate



Cloud Tom

- Tag size is independent of the number of parties
- Aggregation of dynamic subset of parties

KeyGen

AggVer



Scientist Jerry

Our Requirement



Sensor Mice



Cloud Tom



Scientist Jerry

- Tag size is independent of function depth and size
- Tag size is independent of the number of parties
- Verification is independent of the dataset
- Aggregation of dynamic subset of parties

Outline

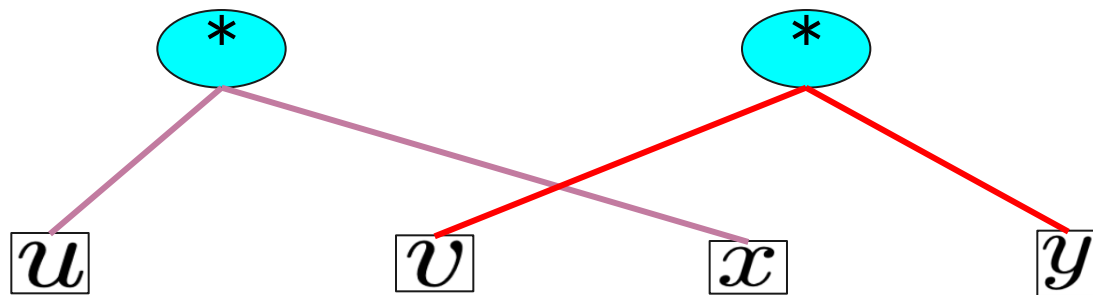


1. Motivating Application
2. Tools Used
3. Why Are These Tools Not Enough ?
4. Syntax
5. Security Definition
6. Construction
7. Summary

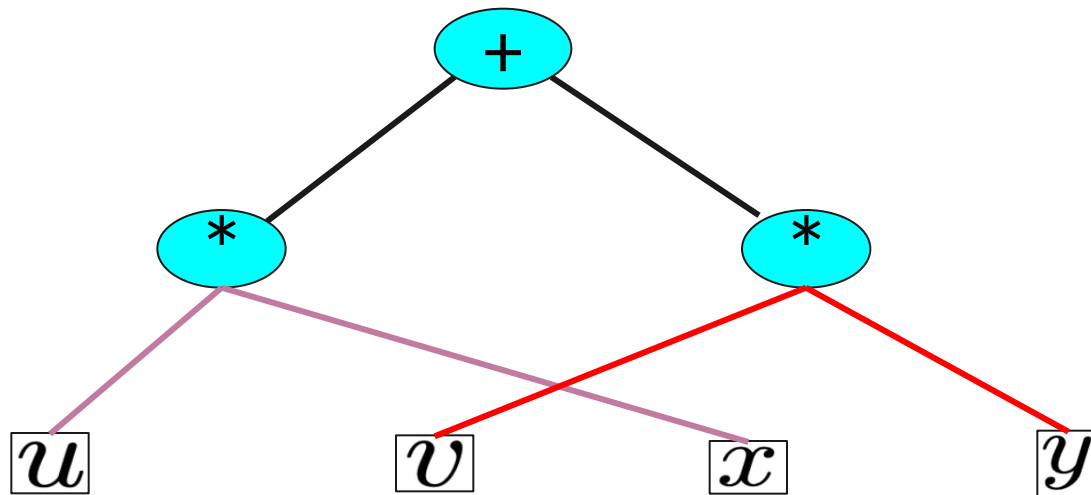
Arithmetic Circuit

 u v x y

Arithmetic Circuit

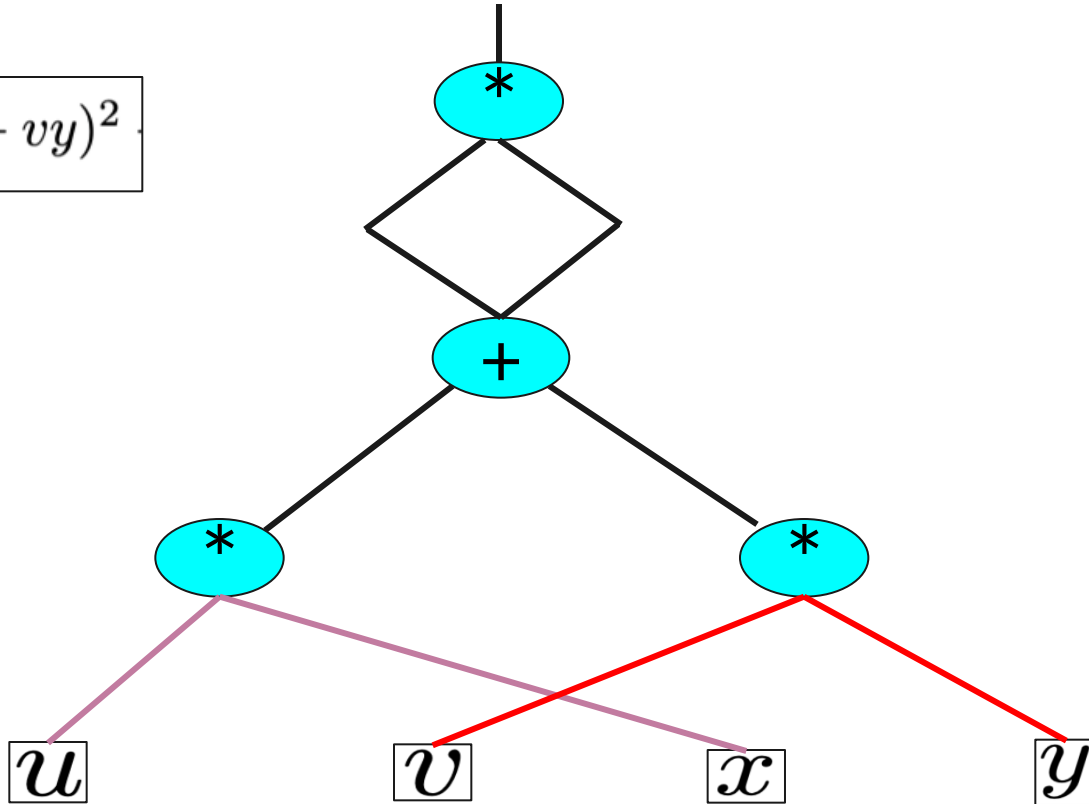


Arithmetic Circuit



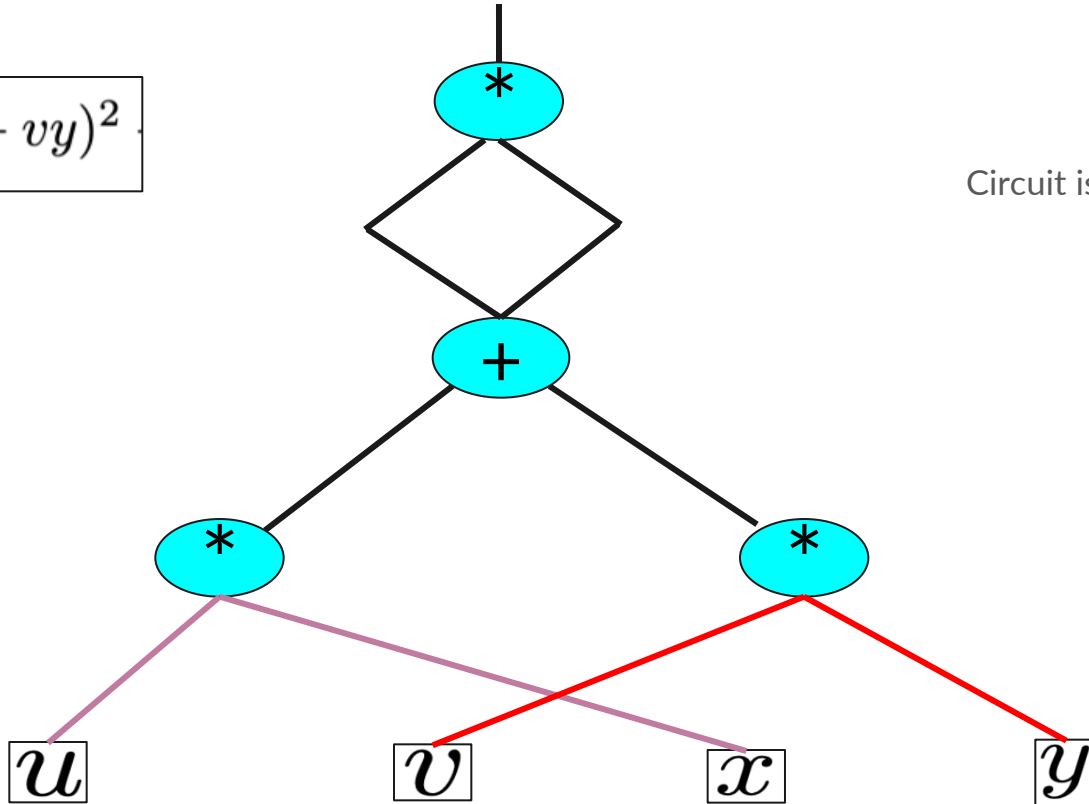
Arithmetic Circuit

$$P_{out} = (ux + vy)^2$$



Arithmetic Circuit

$$P_{out} = (ux + vy)^2$$



Circuit is a **Program**

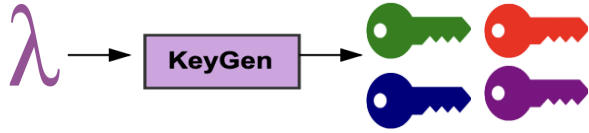
Syntax

σ 

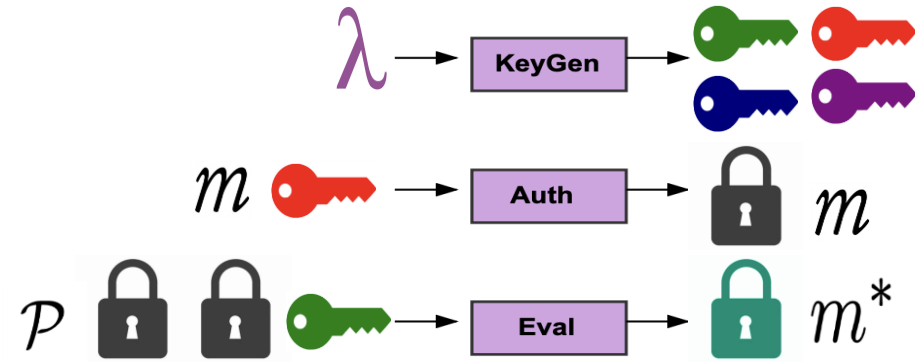
Λ  

σ^* 

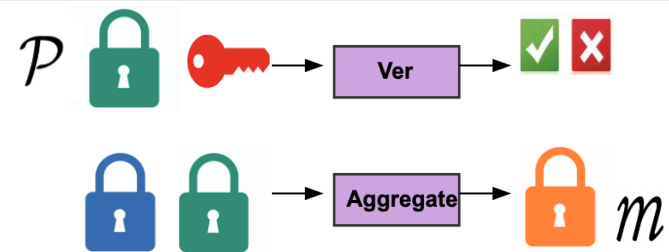
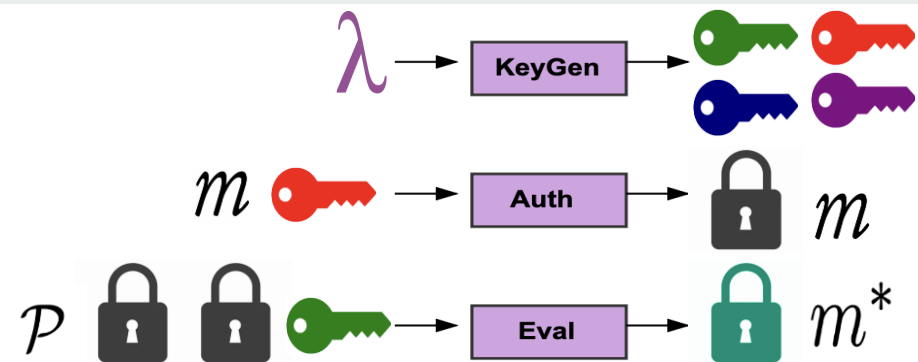
Syntax



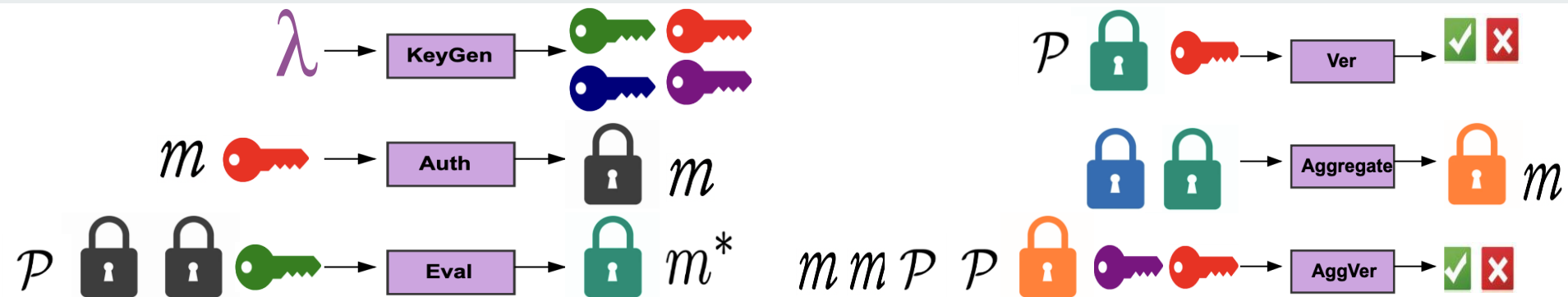
Syntax



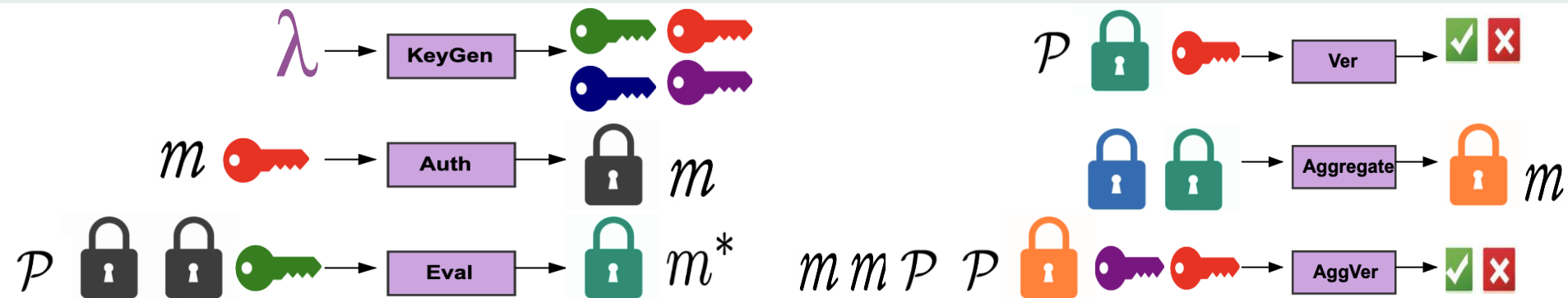
Syntax



Syntax



Syntax



Aggregation Correctness:

$$\Pr[\text{AggVer}((sk_l, m_l, \mathcal{P}_l)_{\forall l \in U}, \sigma^*) = 1] = 1$$

Syntax Summary

2. Auth



Sensor Mice

3. Eval



Cloud Tom

4. Aggregate

1. Keygen



Scientist Jerry

5. AggVer

Outline

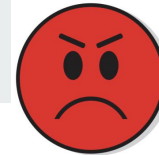


1. Motivating Application
2. Tools Used
3. Why Are These Tools Not Enough ?
4. Syntax
5. Security Definition
6. Construction
7. Summary

Unforgeability



(Challenger)



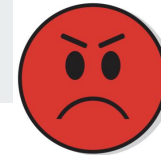
(Adversary)

Unforgeability



(Challenger)

Step 1: Initialize
Create (ek, sk) for all parties



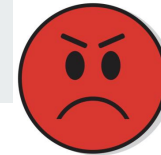
(Adversary)

Step 1: Initialize
Receive ek_1 and (ek, sk) of others

Unforgeability



(Challenger)



(Adversary)

Step 1: Initialize
Create (ek, sk) for all parties

Step 1: Initialize
Receive ek_1 and (ek, sk) of others

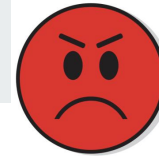
Step 2: Auth(τ, m)
Return σ under sk_1

Step 2: Authenticate Query
Send (τ, m)

Unforgeability



(Challenger)



(Adversary)

Step 1: Initialize
Create (ek, sk) for all parties

Step 1: Initialize
Receive ek_1 and (ek, sk) of others

Step 2: Auth(τ, m)
Return σ under sk_1

Step 2: Authenticate Query
Send (τ, m)

Step 3: ver(m, P, σ) under sk_1
Return 0/1

Step 3: Verification Query
Send (m, P, σ)

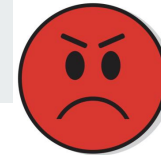
Step 4: AggVer($(m', P')_U, \sigma'$)
Including P_1
Return 0/1

Step 4: AggVer Query
Send $|U|$ number of (m', P') tuples,
 σ'

Unforgeability



(Challenger)



(Adversary)

Step 1: Initialize
Create (ek, sk) for all parties

Step 1: Initialize
Receive ek_1 and (ek, sk) of others

Step 2: Auth(τ, m)
Return σ under sk_1

Step 2: Authenticate Query
Send (τ, m)

Step 3: ver(m, P, σ) under sk_1
Return 0/1

Step 3: Verification Query
Send (m, P, σ)

Step 4: AggVer($(m', P')_{|U}, \sigma'$)
Including P_1
Return 0/1

Step 4: AggVer Query
Send $|U|$ number of (m', P') tuples,
 σ'

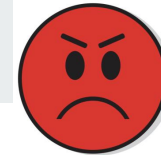
Step 5: Finalise
Run AggVer($(m^*, P^*)_{|U|}, \sigma^*$)
Including P_1
Return 0/1

Step 5: Finalise
Send $|U|$ number of (m^*, P^*) tuples,
 σ^*

Unforgeability



(Challenger)



(Adversary)

- Type 1 Forgery
- Type 2 Forgery

Step 5: Finalise
Run $\text{AggVer}((m^*, P^*)_{|U|}, \sigma^*)$
Including P_1
Return 0/1



Step 5: Finalise
Send $|U|$ number of (m^*, P^*) tuples,
 σ^*

Security Summary



Sensor Mice



Cloud Tom



Scientist Jerry

Security Summary



Sensor Mice



Cloud Tom



Scientist Jerry

Outline

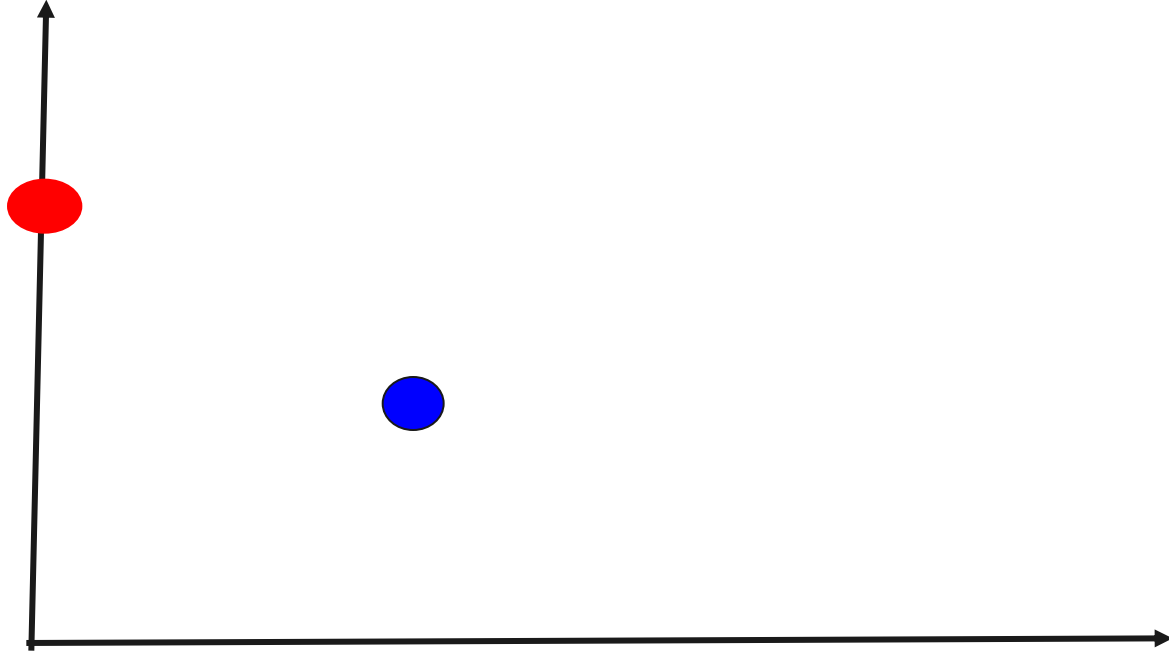


1. Motivating Application
2. Tools Used
3. Why Are These Tools Not Enough ?
4. Syntax
5. Security Definition
- 6. Construction**
7. Summary

Can polynomials give a MAC ? [FC13](#)

m τ

x_0



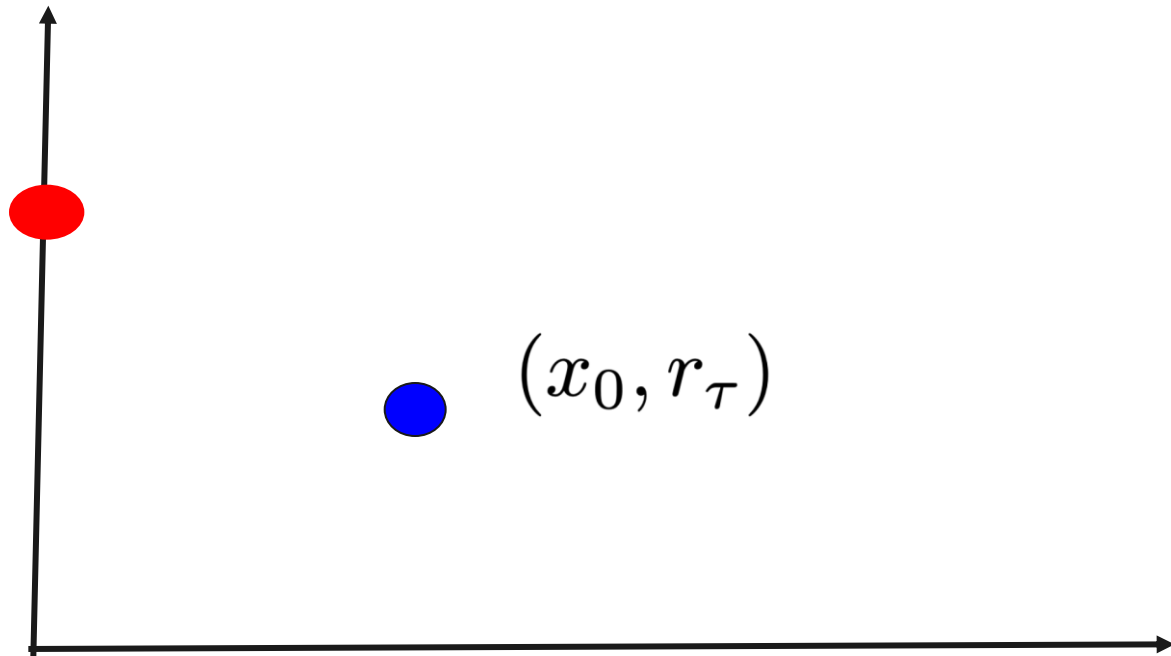
Can polynomials give a MAC ? [FC13](#)

m τ

x_0

$(0, m)$

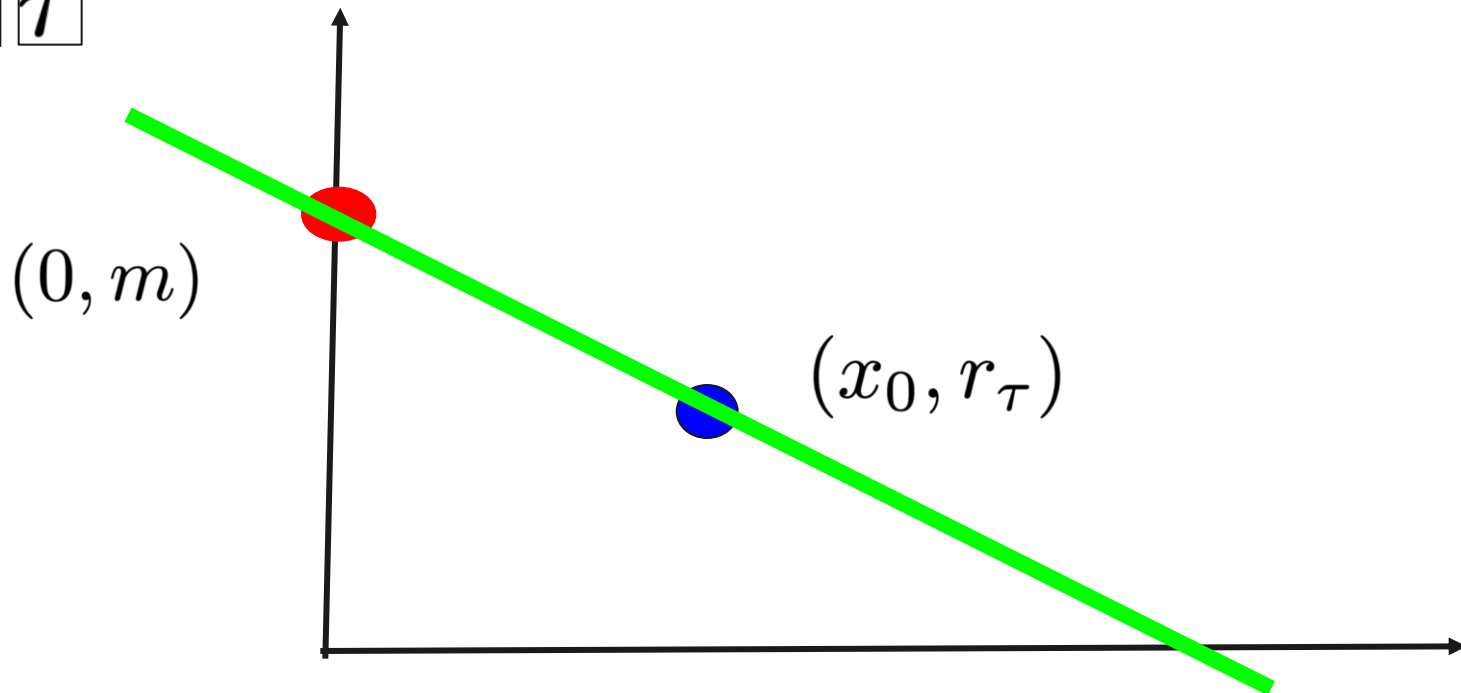
(x_0, r_τ)



Can polynomials give a MAC ? [FC13](#)

m τ

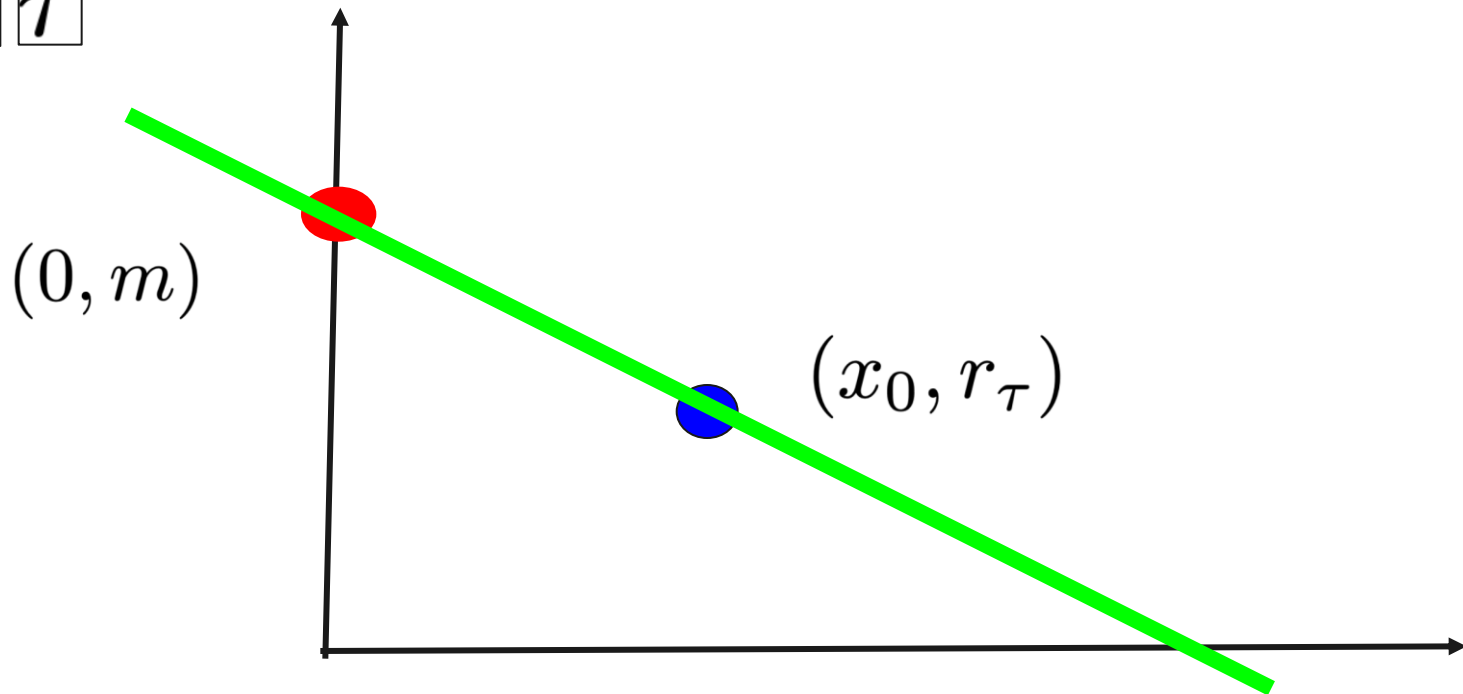
x_0



Can polynomials give a MAC ? [FC13](#)

m τ

x_0

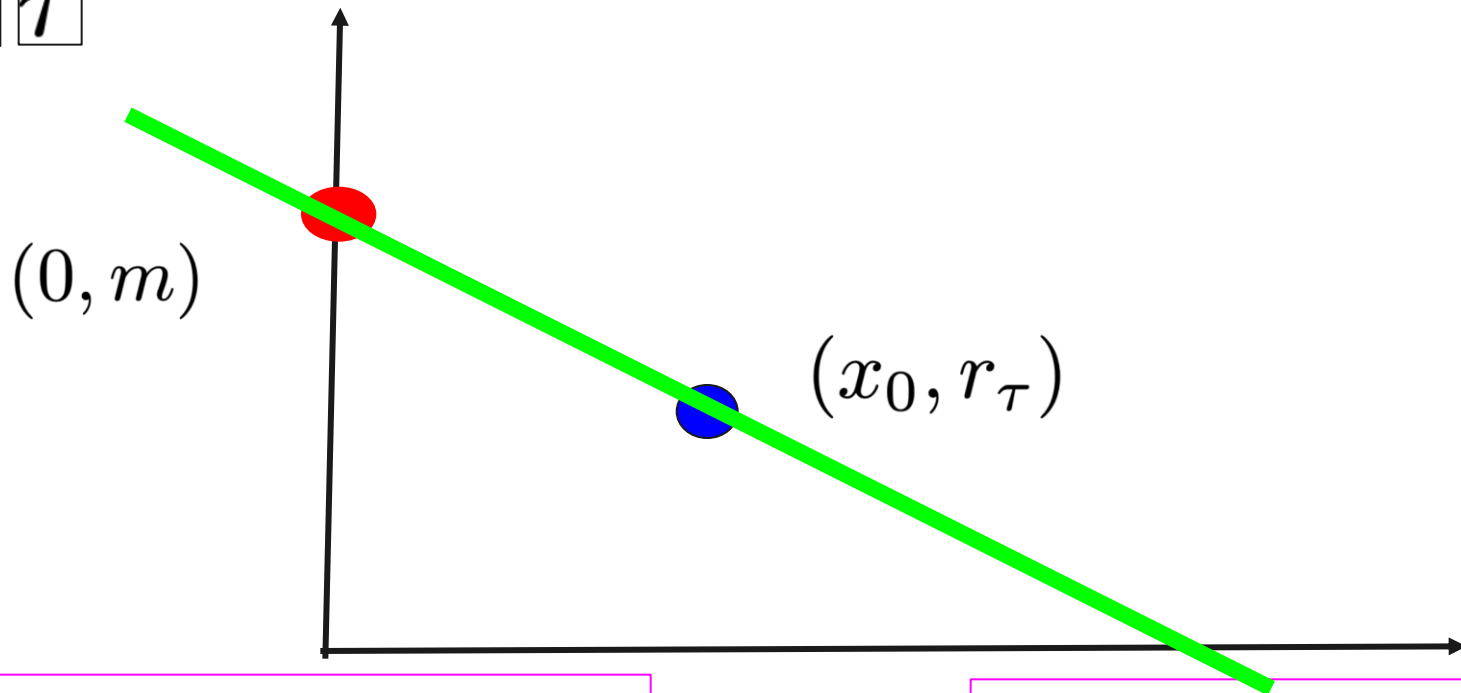


$$y = m + \frac{r_\tau - m}{x_0} \cdot x$$

Can polynomials give a MAC ? [FC13](#)

m τ

x_0



$$y = m + \frac{r_\tau - m}{x_0} \cdot x$$

\equiv

$$\left(m, \frac{r_\tau - m}{x_0} \right)$$

σ



$$\sigma' = (y'_0, y'_1), m'$$



Verification:
Under x_0

$$y'_0 = m'$$

$$y'_0 + y'_1 \cdot x_0 = r_\tau$$

Can polynomials give a Homomorphic MAC ? [FC13](#), [FE16](#), [FA24](#)

$$y = c + ax \quad y \in \mathbb{Z}_p[x]$$

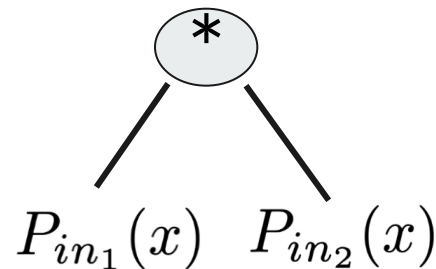
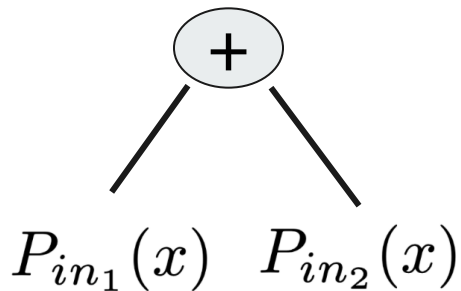
Can polynomials give a Homomorphic MAC ? [FC13](#), [FE16](#), [FA24](#)

$$P_{in_1}(x) = c_1 + a_1 \cdot x$$

$$P_{in_2}(x) = c_2 + a_2 \cdot x$$

$$y = c + ax \quad y \in \mathbb{Z}_p[x]$$

Can polynomials give a Homomorphic MAC ? [FC13](#), [FE16](#), [FA24](#)



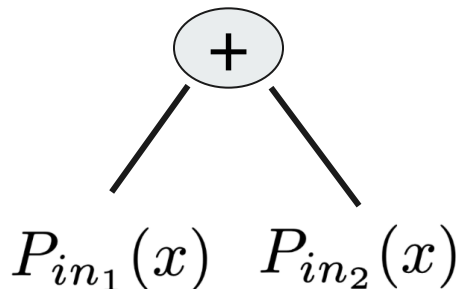
$$P_{in_1}(x) = c_1 + a_1 \cdot x$$

$$P_{in_2}(x) = c_2 + a_2 \cdot x$$

$$y = c + ax \quad y \in \mathbb{Z}_p[x]$$

Can polynomials give a Homomorphic MAC ? [FC13](#), [FE16](#), [FA24](#)

$$\begin{aligned}P_{out}(x) &= P_{in_1}(x) + P_{in_2}(x) \\&= (c_1 + a_1 \cdot x) + (c_2 + a_2 \cdot x) \\&= (c_1 + c_2) + (a_1 + a_2)x\end{aligned}$$



$$P_{in_1}(x) = c_1 + a_1 \cdot x$$

$$P_{in_2}(x) = c_2 + a_2 \cdot x$$

$$y = c + ax \quad y \in \mathbb{Z}_p[x]$$

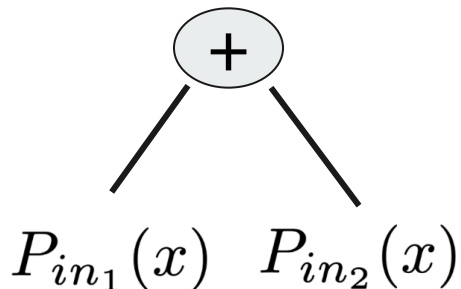
Visualization for additive homomorphism

$$P_{out}(x)$$

$$= P_{in_1}(x) + P_{in_2}(x)$$

$$= (c_1 + a_1 \cdot x) + (c_2 + a_2 \cdot x)$$

$$= (\textcolor{violet}{c}_1 + \textcolor{blue}{c}_2) + (\textcolor{green}{a}_1 + \textcolor{red}{a}_2)x$$



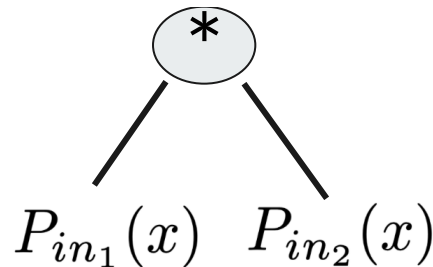
$$P_{in_1}(x) = \textcolor{violet}{c}_1 + \textcolor{green}{a}_1 \cdot x$$

$$P_{in_2}(x) = \textcolor{blue}{c}_2 + \textcolor{red}{a}_2 \cdot x$$

$$y = c + ax \quad y \in \mathbb{Z}_p[x]$$

Can polynomials give a Homomorphic MAC ? [FC13](#), [FE16](#), [FA24](#)

$$\begin{aligned}P_{out}(x) &= P_{in_1}(x) \cdot P_{in_2}(x) \\&= (a_1 \cdot x + c_1) + (a_2 \cdot x + c_2) \\&= (c_1 \cdot c_2) + (a_1 \cdot c_2)x + (a_2 \cdot c_1)x + (a_1 \cdot a_2)x^2\end{aligned}$$



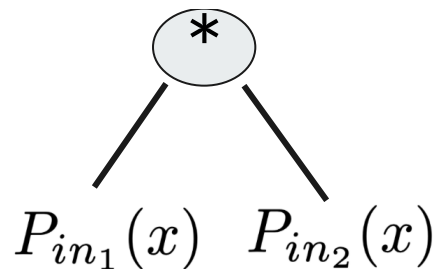
$$P_{in_1}(x) = c_1 + a_1 \cdot x$$

$$P_{in_2}(x) = c_2 + a_2 \cdot x$$

$$y = c + ax \quad y \in \mathbb{Z}_p[x]$$

Visualization for multiplicative homomorphism

$$\begin{aligned}P_{out}(x) &= P_{in_1}(x) \cdot P_{in_2}(x) \\&= (a_1 \cdot x + c_1) + (a_2 \cdot x + c_2) \\&= (\textcolor{violet}{c}_1 \cdot \textcolor{blue}{c}_2) + (\textcolor{green}{a}_1 \cdot \textcolor{blue}{c}_2)x + (\textcolor{red}{a}_2 \cdot \textcolor{violet}{c}_1)x + (\textcolor{green}{a}_1 \cdot \textcolor{red}{a}_2)x^2\end{aligned}$$



$$P_{in_1}(x) = \textcolor{violet}{c}_1 + \textcolor{green}{a}_1 \cdot x$$

$$P_{in_2}(x) = \textcolor{blue}{c}_2 + \textcolor{red}{a}_2 \cdot x$$

$$y = c + ax \quad y \in \mathbb{Z}_p[x]$$

Unravelling Multiplication

$$P_{out}(x) = m_1 \cdot m_2 + m_2 \cdot \left(\frac{r_{\tau_1} - m_1}{x_0} \right) \cdot x + m_1 \cdot \left(\frac{r_{\tau_2} - m_2}{x_0} \right) \cdot x + \left(\frac{r_{\tau_1} - m_1}{x_0} \right) \cdot \left(\frac{r_{\tau_2} - m_2}{x_0} \right) \cdot x^2$$

$$P_{out}(0) = m_1 \cdot m_2 \bmod p$$

$$P_{out}(x_0) = r_{\tau_1} \cdot r_{\tau_2} \bmod p$$

Correctness of Homomorphic MAC^{FC13}



$$\sigma' = (y'_0, \dots, y'_d), m'$$



Verification :

Under x_0

$$y'_0 = m'$$
$$\sum_{k=0}^d y'_k \cdot x_0^k = f(r_{\tau_1}, \dots, r_{\tau_n})$$

Can Polynomials give Multi Key Homomorphic Aggregate MAC ?

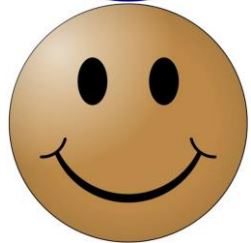
Naive Attempt:



$ek_1, sk_1, m_1, \Lambda_1$



$ek_2, sk_2, m_2, \Lambda_2$



$ek_{|U|}, sk_{|U|}, m_{|U|}, \Lambda_{|U|}$

Λ Succinct Homomorphic MAC

$$\Lambda_1 || \Lambda_2 || \dots || \Lambda_{|U|}$$

Issue: Efficiency

Can Polynomials give Multi Key Homomorphic Aggregate MAC ?

Attempt 2:

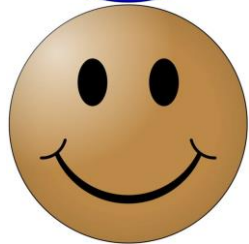
Λ Succinct Homomorphic MAC



$ek_1, sk_1, m_1, \Lambda_1$



$ek_2, sk_2, m_2, \Lambda_2$



$ek_{|U|}, sk_{|U|}, m_{|U|}, \Lambda_{|U|}$

$$\Lambda_1 + \Lambda_2 + \dots + \Lambda_{|U|}$$

Issue: Privacy

Can Polynomials give Multi Key Homomorphic Aggregate MAC ?

Attempt 3:



$ek_1, sk_1, m_1, \Lambda_1, \mathcal{H}(m_1)$



$ek_2, sk_2, m_2, \Lambda_2, \mathcal{H}(m_2)$



$ek_{|U|}, sk_{|U|}, m_{|U|}, \Lambda_{|U|}, \mathcal{H}(m_{|U|})$

Λ Succinct Homomorphic MAC
 \mathcal{H} Hash Function

$$\sigma^* = \sum_{l \in U} \mathcal{H}(m_l) \cdot \Lambda_l$$

(Informally)

Correctness



$\sigma^*, m_1^*, \dots, m_{|U|}^*$



Verification(Informal) : check

Under $(sk_1, \dots, sk_{|U|})$

$$\sigma^* = \sum_{l \in U} sk_l \cdot \mathcal{H}(m_l^*) \cdot f_l(r_{\tau_{l,1}}, \dots, r_{\tau_{l,n}}) + \sum_{l \in U} sk_l \cdot \mathcal{H}(m_l^*) \cdot m_l^*$$

Intuition for the Proof

Theorem 1. *(Informal) If co-CDH is (t', ϵ') hard over groups $(\mathcal{G}_1, \mathcal{G}_2)$ and PRF is secure, then HA-MAC scheme is (t, Q, ϵ) secure in the random oracle model for all t, ϵ satisfying*

$$\epsilon < \frac{Q}{2^\lambda} + \epsilon' \quad \text{and} \quad t > t'$$

where Q is the number of queries and λ is the security parameter.

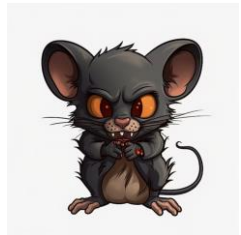
The proof relies loosely on a variant of **Schwartz Zippel**^{[Zp](#)} and a reduction to **co-CDH**^{[B13](#)} assumption through a series of hybrids and is secure in the **Random Oracle Model**.

Outline



1. Motivating Application
2. Tools Used
3. Why Are These Tools Not Enough ?
4. Syntax
5. Security Definition
6. Construction
7. Summary

Summary



Sensor Mice




Cloud Tom



Scientist Jerry

- Our primitive enables an **untrusted** server to produce a short certificate to prove that he has performed correct (**disjoint**) computations on multiple users' data.
- The **size of this proof** is independent of the number of users and the complexity of the performed computations.
- We give a construction of this primitive based on the **co-CDH** assumption in the **random oracle model**

Open Problems:

- 
- Remove ROM
 - Independence of Verification from the complexity of function
 - Independence of the size of keys from the depth of the function.



Thank You!



References:

1. Catalano, D., & Fiore, D. (2013, May). Practical homomorphic MACs for arithmetic circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 336-352). Berlin, Heidelberg: Springer Berlin Heidelberg.
2. Gennaro, R., & Wichs, D. (2013, December). Fully homomorphic message authenticators. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 301-320). Berlin, Heidelberg: Springer Berlin Heidelberg.
3. Boneh, D., Gentry, C., Lynn, B., & Shacham, H. (2003). Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22* (pp. 416-432). Springer Berlin Heidelberg.
4. Fiore, D., Mitrokotsa, A., Nizzardo, L., & Pagnin, E. (2016, November). Multi-key homomorphic authenticators. In *International conference on the theory and application of cryptology and information security* (pp. 499-530). Berlin, Heidelberg: Springer Berlin Heidelberg.
5. Anthoine, G., Balbás, D., & Fiore, D. (2024, August). Fully-succinct multi-key homomorphic signatures from standard assumptions. In *Annual International Cryptology Conference*(pp. 317-351). Cham: Springer Nature Switzerland.
6. Zippel, R. (1979, June). Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation* (pp. 216-226). Berlin, Heidelberg: Springer Berlin Heidelberg.