Threshold Cryptosystems from Threshold Fully Homomorphic Encryption

Aayush Jain, UCLA

AUTHORS:

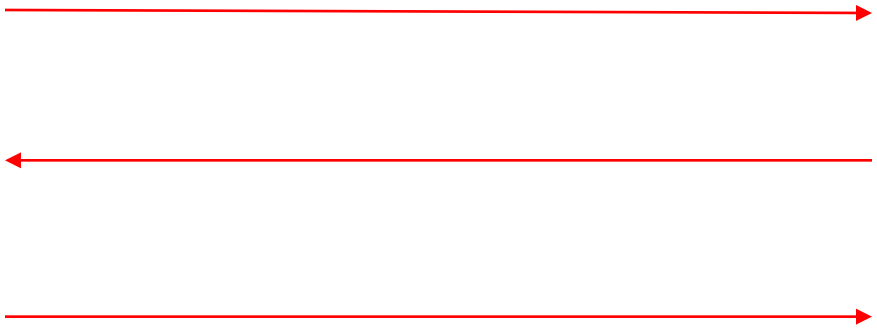
DAN BONEH, ROSARIO GENNARO, STEVEN GOLDFEDER, AAYUSH JAIN, SAM KIM, PETER M. R. RASMUSSEN AND AMIT SAHAI

Introduction to Characters

Tony Stark: Good Guy

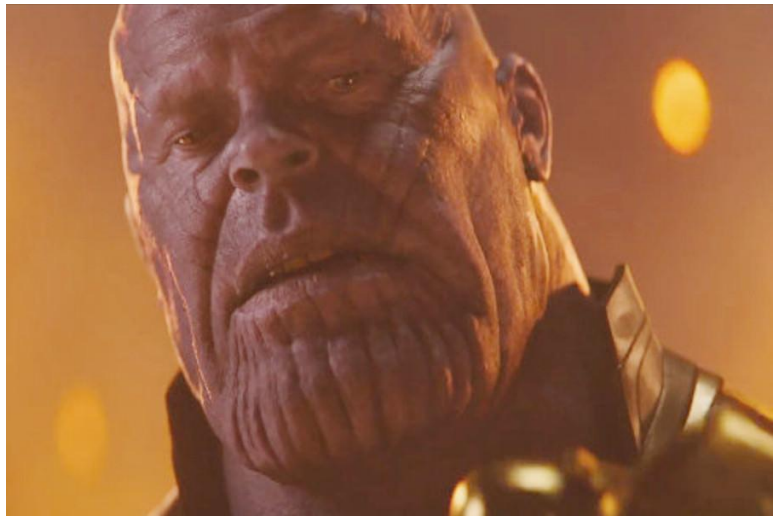
Thanos: Bad Guy

Key Management



For security, need to have private information.

Key Management



KeyGen(1^λ) →

Key Management

KeyGen(1^λ) →

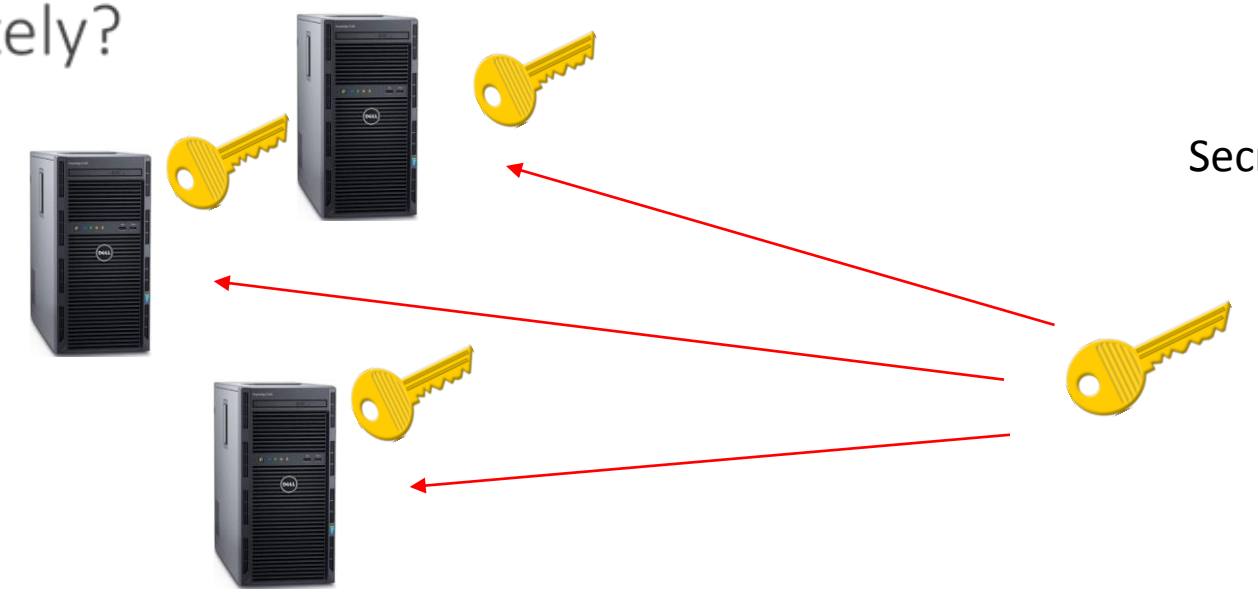
Key Management is prone to side channel leaks, social hacking, human error etc.

Main Question

Can we address this issue at more fundamental level?

Threshold Cryptography

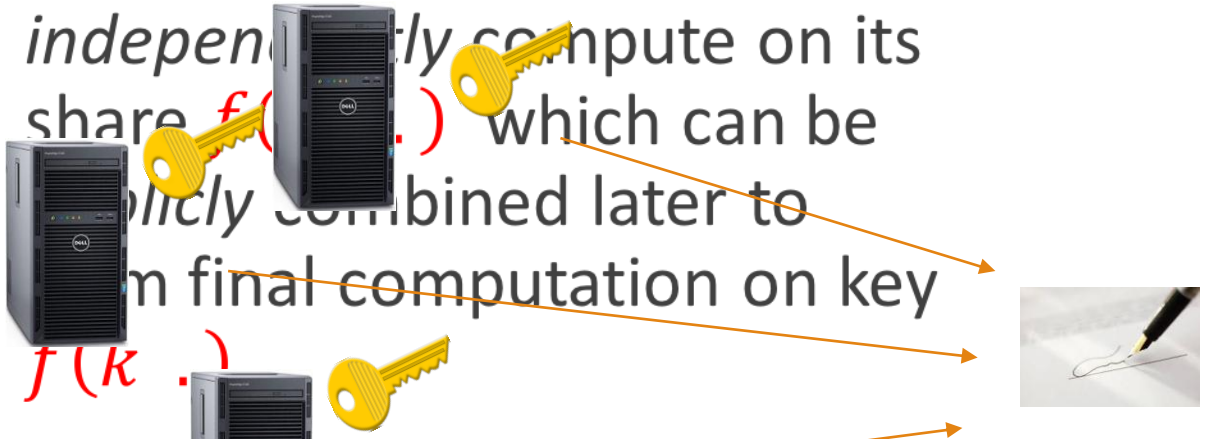
Naïve Solution: Can we divide key k into shares $s_1, \ldots, s_N$ and store them separately?



Secret Sharing

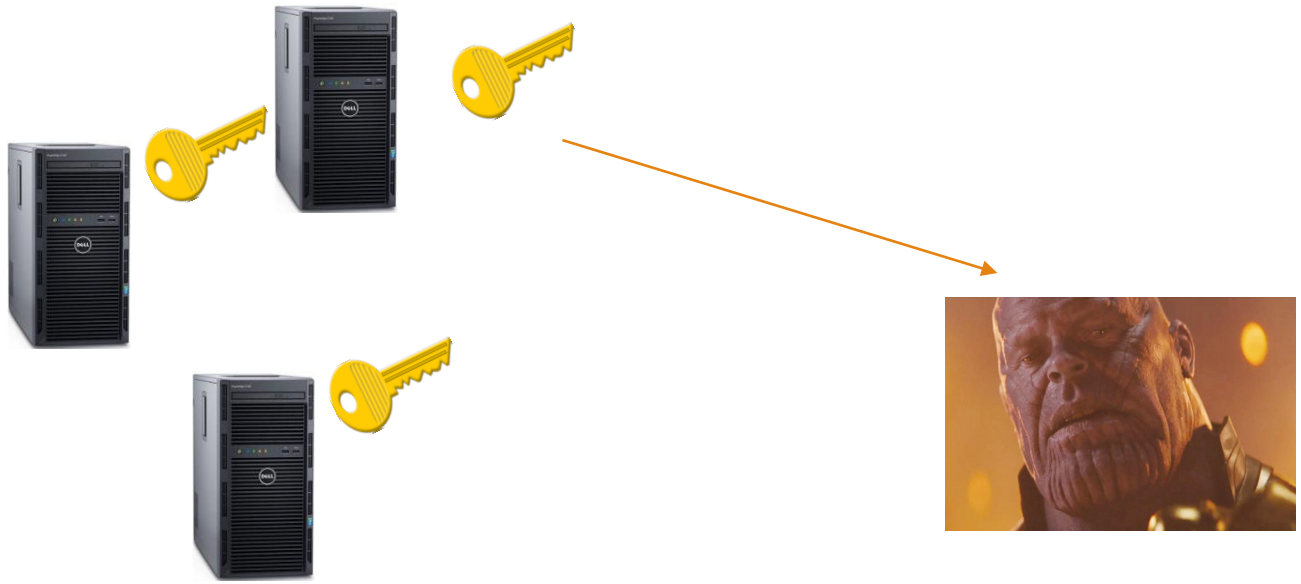
Threshold Cryptography

- **Correctness:** Each server can *independently* compute on its share $f(\ .\ )$ which can be *publicly* combined later to form final computation on key $f(k\ .\ )$
- **Security:** Hard to form final computation $f(k\ .\ )$ without all key shares $s_1, \ldots . s_N$

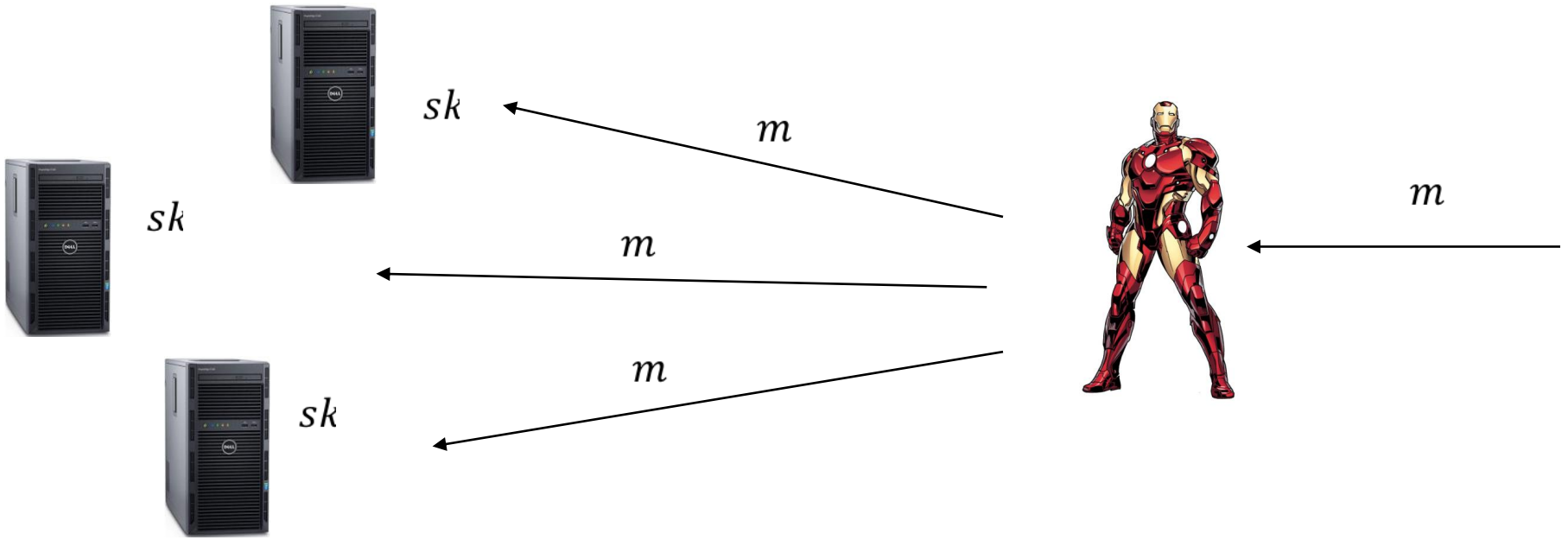


Threshold Cryptography (t out of n)

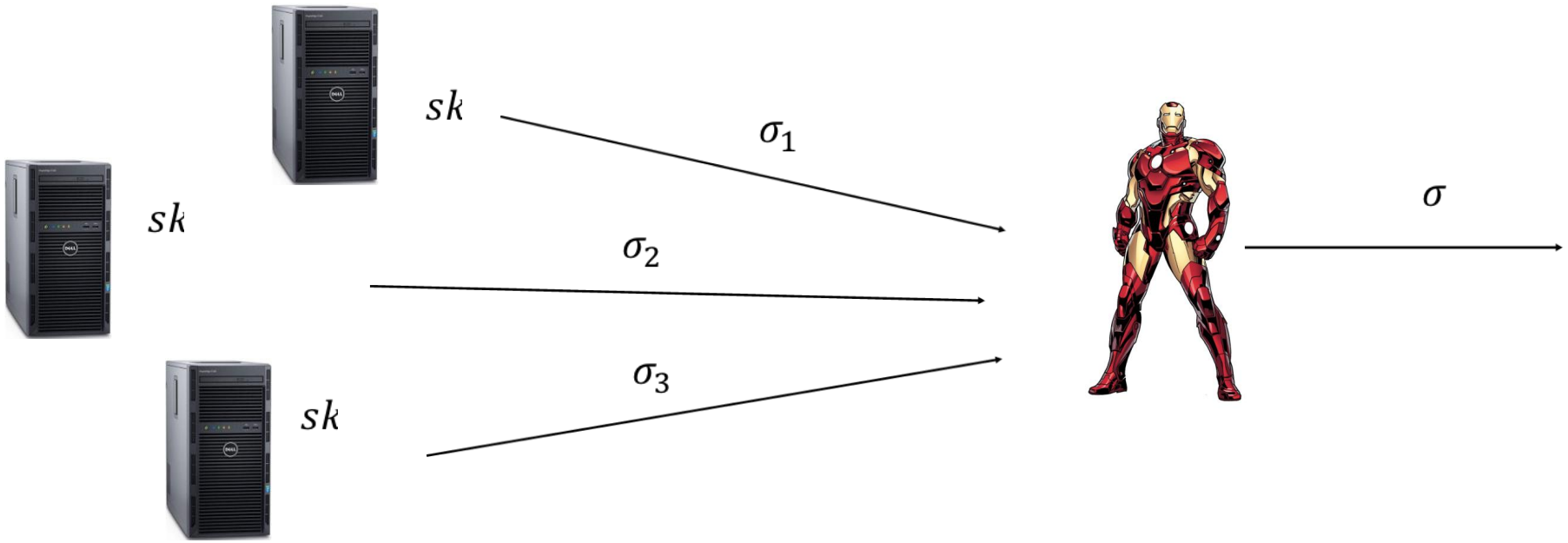
- **Correctness:** Each server can *independently* compute on its share $f(s_i, .)$. Any **t** evaluation shares $f(s_i, .)$ can be *publicly* combined later to form final computation $f(k .)$
- **Security:** Hard to form final computation $f(k .)$ without **t** key shares



Threshold Signatures

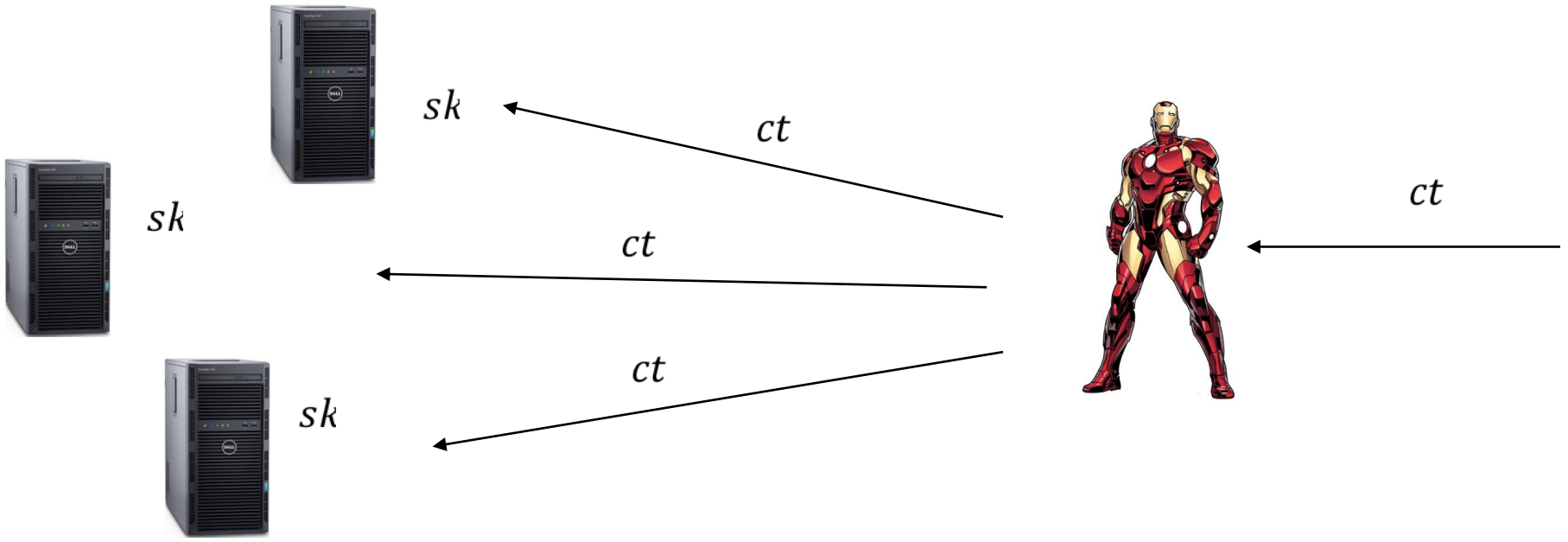


Threshold Signatures

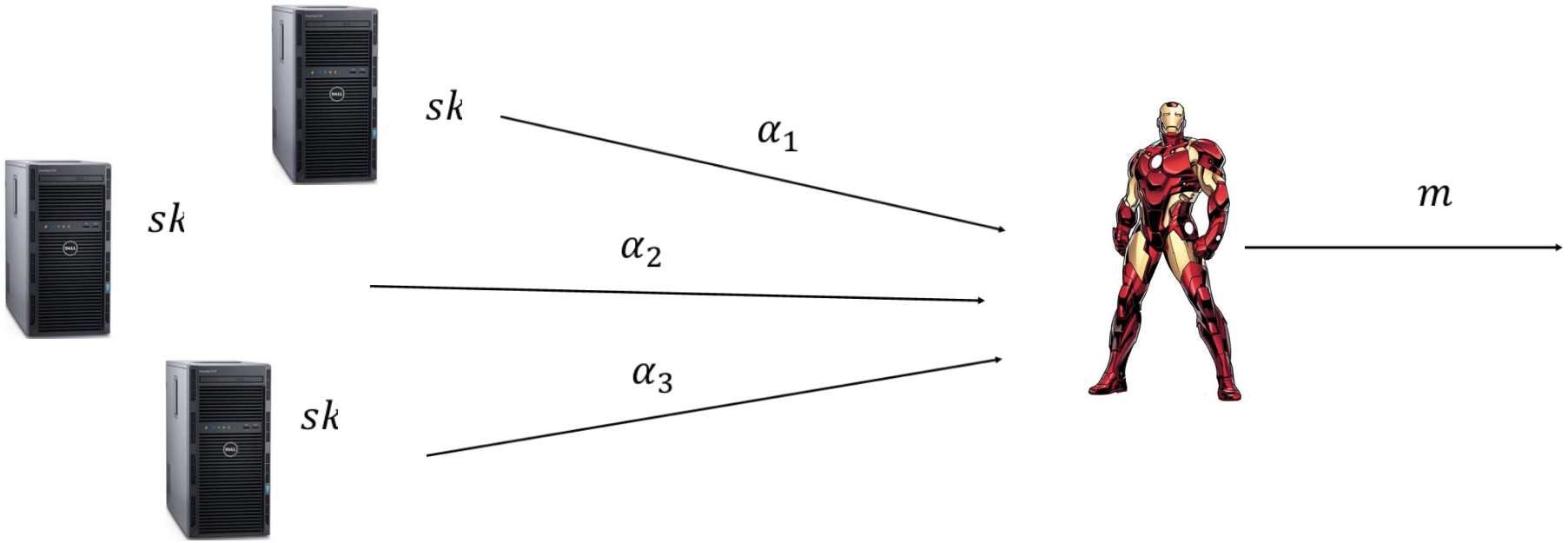


Requirements: Unforgeability, Compactness, Correctness, Robustness etc..

Threshold Public Key Encryption



Threshold Public Key Encryption



Requirements: CCA Security, Compactness, Correctness, Robustness etc..

Related Works

- RSA Signatures [Fra89, DDFY94, GRJK07, Sho00]
- Schnorr Signatures [SS01]
- (EC)DSA Signatures [GJKR01, GGN16]
- BLS Signatures [BLS04, Bol03]
- Cramer-Shoup Encryption [CG99]
- Many More [SG02, DK05, BBH06,…]

Our Results

- Construct Threshold Fully Homomorphic Encryption (TFHE)
- Formalised the concept of Universal Thresholdizer (UT).
- Show how to use UT as a general tool for constructing threshold cryptosystems
- Construct UT from TFHE.
- New Constructions for a variety of threshold cryptosystems: Threshold Signatures, CCA secure PKE, distributed PRFs, Function Secret Sharing from LWE

~~Threshold~~ Fully Homomorphic Encryption

- $Setup(1^{\lambda}) \longrightarrow (pk, sk)$
- $Encrypt(pk, m) \longrightarrow ct$
- $Eval(pk, C, ct_1, \ldots, ct_k) \longrightarrow ct_{eval}$
- $Decrypt(sk, ct_{eval}) \longrightarrow m_{eval} = C(m_1, \ldots, m_k)$

Threshold Fully Homomorphic Encryption (TFHE)

- $Setup(1^{\lambda}, N, t) \longrightarrow (pk, sk_1, \dots, sk_N)$
- $Encrypt(pk, m) \longrightarrow ct$
- $Eval(pk, C, ct_1, \dots, ct_k) \longrightarrow ct_{eval}$
- $PartDecrypt(sk_i, ct_{eval}) \longrightarrow p_i$
- $FinDecrypt(pk, \{p_i\}) \longrightarrow m_{eval}$

Correctness:

For any circuit $C: \{0,1\}^k \to \{0,1\}$, $ct_{eval} \leftarrow Eval(pk, C, ct_1, \dots, ct_k)$, $|S| \geq t$, $|p_i| \leq poly(\lambda, N, d)$

$$FinDec(pk, \{PartDecrypt(sk_i, ct_{eval})\}_{i \in S}) = C(m_1, \dots, m_k)$$

Security Definitions

Semantic Security:

Adversary is given $z = \{sk_i\}_{i \in S}$ where $|S| < t$. Then for any m_0, m_1

$$(Encrypt(pk, m_0), z) \approx (Encrypt(pk, m_1), z)$$

Simulation Security:

There exists a simulator Sim such that given $z = \{sk_i\}_{i \in S}$ where $|S| = t\text{-}1$.
Then for any ct_{eval} and any $i \in N \setminus S$

$$(Sim(i, z, m_{eval}, ct_{eval})) \approx_{stat} (\text{PartDec}(sk_i, ct_{eval}))$$

Starting Point: [GSW13] FHE Scheme

- Ciphertext ct is a matrix in $\{0,1\}^{l \times l}$.
- Secret key $\vec{s}$ is a vector in $\mathbb{Z}_q^l$. It has the following structure.

$$\vec{s} = (s_1, \ldots, s_{l-1}, \lfloor \frac{q}{2} \rfloor)$$

- **(Approximate Eigenvector Property).** For any encryption ct,

$$\vec{s} \cdot ct = m\vec{s} + noise$$

Recap: [GSW13]

- Recall,

$$\vec{s} \cdot ct = m\vec{s} + noise$$

- **Homomorphic Addition:** $ct_1 + ct_2$. Observe that,

$$\vec{s} \cdot ct_1 + \vec{s} \cdot ct_2 = (m_1 + m_2)\vec{s} + noise$$

- **Homomorphic Multiplication:** $ct_1 \cdot ct_2$. Observe that,

$$\vec{s} \cdot ct_1 \cdot ct_2 = m_1 \cdot m_2 \cdot \vec{s} + noise$$

Recap: [GSW13]

- Recall,

$$\vec{s} \cdot ct = m\vec{s} + noise \qquad \vec{s} = (s_1, ..., s_l, \lfloor \frac{q}{2} \rfloor)$$

Decryption is linear in $\vec{s}$!

- $Decrypt(\vec{s}, ct) =$

$$\begin{aligned} &\vec{s} \cdot ct \cdot [0, ..., 0, 1]^T \\ &= m \cdot \vec{s} \cdot [0, ..., 0, 1]^T + noise \cdot [0, ..., 0, 1]^T \\ &= m\lfloor \frac{q}{2} \rfloor + noise' \end{aligned}$$

Very First Observation

- Shamir Secret Share $\vec{s} \rightarrow \vec{s}_1, \ldots, \vec{s}_N$

- Thus, for every set S, with $|S| = t$, and $i \in S$, there exists lagrange coefficient λ_i such that:

$$\vec{s} = \Sigma_{i \in S} \lambda_i \vec{s}_i$$

Initial Idea

- Define partial decryption:

$$PartDec(pk, s_i, ct) = s_i \cdot ct \cdot [0, \dots, 0,1]^T$$

- Then, final decryption as

$$FinDec(pk, s_i \cdot ct \cdot [0, \dots, 0,1]^T_{i \in S}) = \sum \lambda_i s_i \cdot ct \cdot [0, \dots, 0,1]^T$$

$$= s \cdot ct \cdot [0, \dots, 0,1]^T$$

$$= m \left\lfloor \frac{q}{2} \right\rfloor + noise$$

Noise leaks too much information (in form of linear equations), and leads to attacks!

FHE decryption should just reveal message

Smudging with noise

- Define partial decryption:

$$PartDec(pk, s_i, ct) = s_i \cdot ct \cdot [0, \dots, 0,1]^T + noise_i$$

Correctness is lost!

- Then, final decryption as

$$FinDec\left(pk, s_i \cdot ct \cdot [0, \dots, 0,1]^T{}_{i \in S}\right) = \sum_{i \in S} \lambda_i \left(s_i \cdot ct \cdot [0, \dots, 0,1]^T + noise_i\right)$$

$$= s \cdot ct \cdot [0, \dots, 0,1]^T + \sum_{i \in S} \lambda_i noise_i$$

$$= \mathrm{m} \left\lfloor \frac{q}{2} \right\rfloor + noise + BIG$$

How to Fix Noise Blowup?

- Define a new linear secret sharing scheme with low-norm reconstruction coefficients.
- Two ways of doing that:

1. A general purpose secret sharing scheme supporting broader access patterns.
2. More direct modification of Shamir Secret Sharing scheme leading to shorter keys, albeit slightly larger ciphertexts.

{0,1}-LSSS

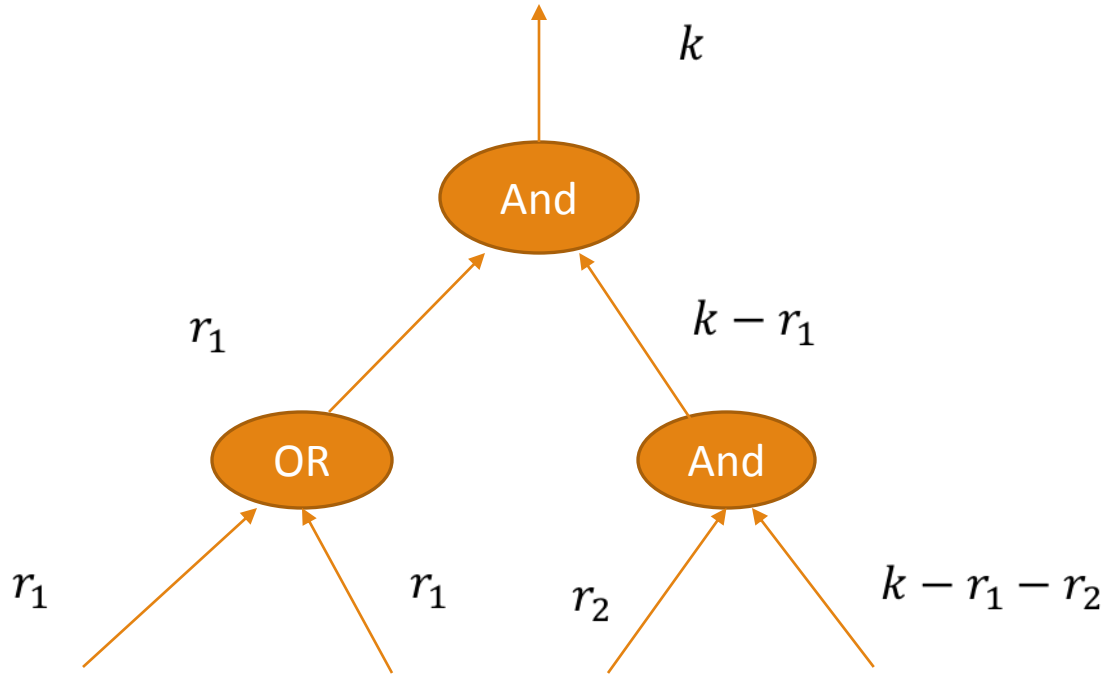
Linear secret sharing scheme for $k \in \mathbb{Z}_q$

- Share$(k, \phi) \rightarrow (s_1, \ldots, s_N) \in \mathbb{Z}_q^n$
- Combine$(\{s_i\}_{i \in S})$:
  - For any set S $with$ $\phi(S) = 1$, there exists efficiently computable coefficients $c_i \in \mathbb{Z}_q$ such that,

$$k = \sum_{i \in S} c_i \cdot s_i$$

- Define {0,1}-LSSS as a class of linear secret sharing schemes where the reconstruction coefficients are always binary.

How Expressive is {0,1}-LSSS?



- $s_1 = r_1$
- $s_2 = r_1$
- $s_3 = r_2$
- $s_4 = k - r_1 - r_2$

$$s = s_1 + s_3 + s_4$$

$$s = s_2 + s_3 + s_4$$

How Expressive is {0,1}-LSSS

- Monotone Boolean Formulas
- [Val84] showed that threshold function can be expressed by a monotone boolean formula
- Note: In {0,1}-LSSS, the reconstruction coefficients λ_i are either 0 or 1

Recap

- Define partial decryption:

$$PartDec(pk, s_i, ct) = s_i \cdot ct \cdot [0, \dots, 0,1]^T + noise_i$$

- Then, final decryption as

$$FinDec(pk, s_i \cdot ct \cdot [0, \dots, 0,1]^T_{i \in S}) = \sum_{i \in S} \lambda_i (s_i \cdot ct \cdot [0, \dots, 0,1]^T + noise_i)$$

Correctness is not lost! Needs careful Security Analysis

$$= s \cdot ct \cdot [0, \dots, 0,1]^T + \sum_{i \in S} noise_i \, \lambda_i$$

$$= \mathrm{m} \left\lfloor \frac{q}{2} \right\rfloor + noise + SMALL$$

More direct way

- Modify the scheme using Clearing the Denominators trick [Sho00,ABVVW12].
- Basic idea is that for any lagrange coefficient $\lambda \in \mathbb{Z}_q$,

$$\lambda N! = O(N!^2)$$

and

$$\lambda^{-1} N! = \mathrm{O}(N!^2)$$

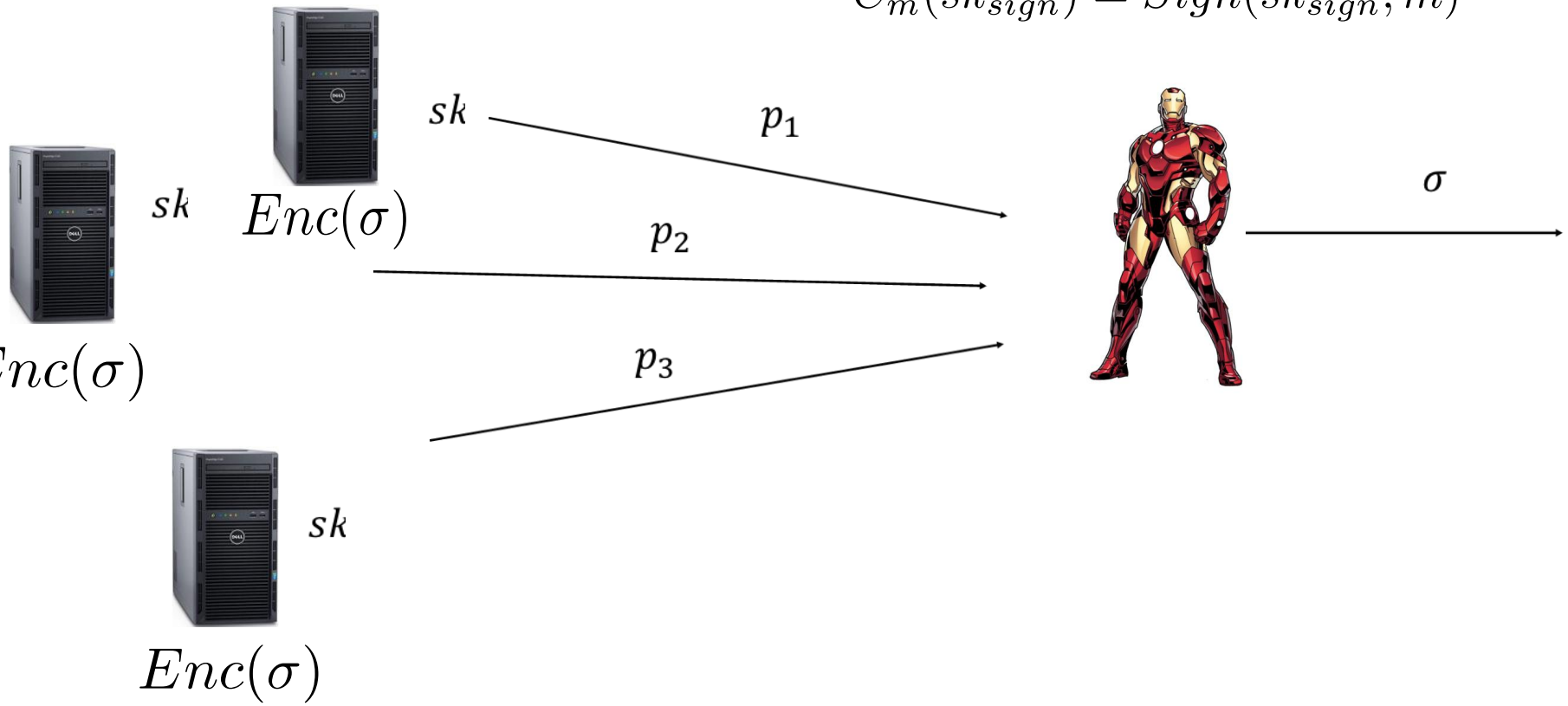
Comparison of two schemes

	Ciphertext /Public Key Size	Key Size/Partial Decryption Size	Access Structure
{0,1}-LSSS Scheme	$poly(\lambda, d)$	$N^{4.2}poly(\lambda, d)$ (for threshold access structure)	Monotone Boolean Formulas
Clearing Denominators	$Npoly(\lambda, d)$	$Npoly(\lambda, d)$	Threshold Access Structures

Threshold Signatures

$pp = (pk, Enc(sk_{sign}))$

$C_m(sk_{sign}) = Sign(sk_{sign}, m)$



Universal Thresholdizer

- Setup$(1^{\lambda}, t, N, \mathrm{x}) \rightarrow (pp, s_1, \ldots, s_N)$
  - TFHE.Setup$(1^{\lambda}, t, N) \rightarrow (pk, sk_1, \ldots, sk_n)$
  - Encrypt(pk,x)$\rightarrow ct$

    pp=(pk, ct) $\qquad\qquad$ $s_i = sk_i$
- Eval(pp,$s_i, C) \rightarrow p_i$
  - $Eval(pk, C, ct) \rightarrow \widehat{ct}$
  - $PartDec(s_i, \widehat{ct}) \rightarrow p_i$
- Combine$(pp, \{p_i\}) \rightarrow C(x)$
  - FinDec(pk,$\{p_i\}) \rightarrow C(x)$

Our Results

- Construct Threshold Fully Homomorphic Encryption (TFHE)
- Formalised the concept of Universal Thresholdizer (UT).
- Show how to use UT as a general tool for constructing threshold cryptosystems
- Construct UT from TFHE.
- New Constructions for a variety of threshold cryptosystems: Threshold Signatures, CCA secure PKE, distributed PRFs, Function Secret Sharing from LWE

Application of Techniques

- Lazy MPC [BJMS18]: An MPC where honest parties can ``go to sleep”- limited computing power, lost connection etc..
- Theoretical Outcome: First MPC with Guaranteed Output Delivery in the standard model in three rounds (Concurrent with [ACGJ18]).
- Amplification: Given an FE/iO candidate with partial security, output a fully secure candidate. Appeared in [AJKS18]

Open Problems

- Not relying on FHE? (More efficient construction)
- More applications
- Better assumptions? (polynomial approximation factor)