Threshold Linearly Homomorphic Encryption on $\mathbb{Z}/2^k\mathbb{Z}$

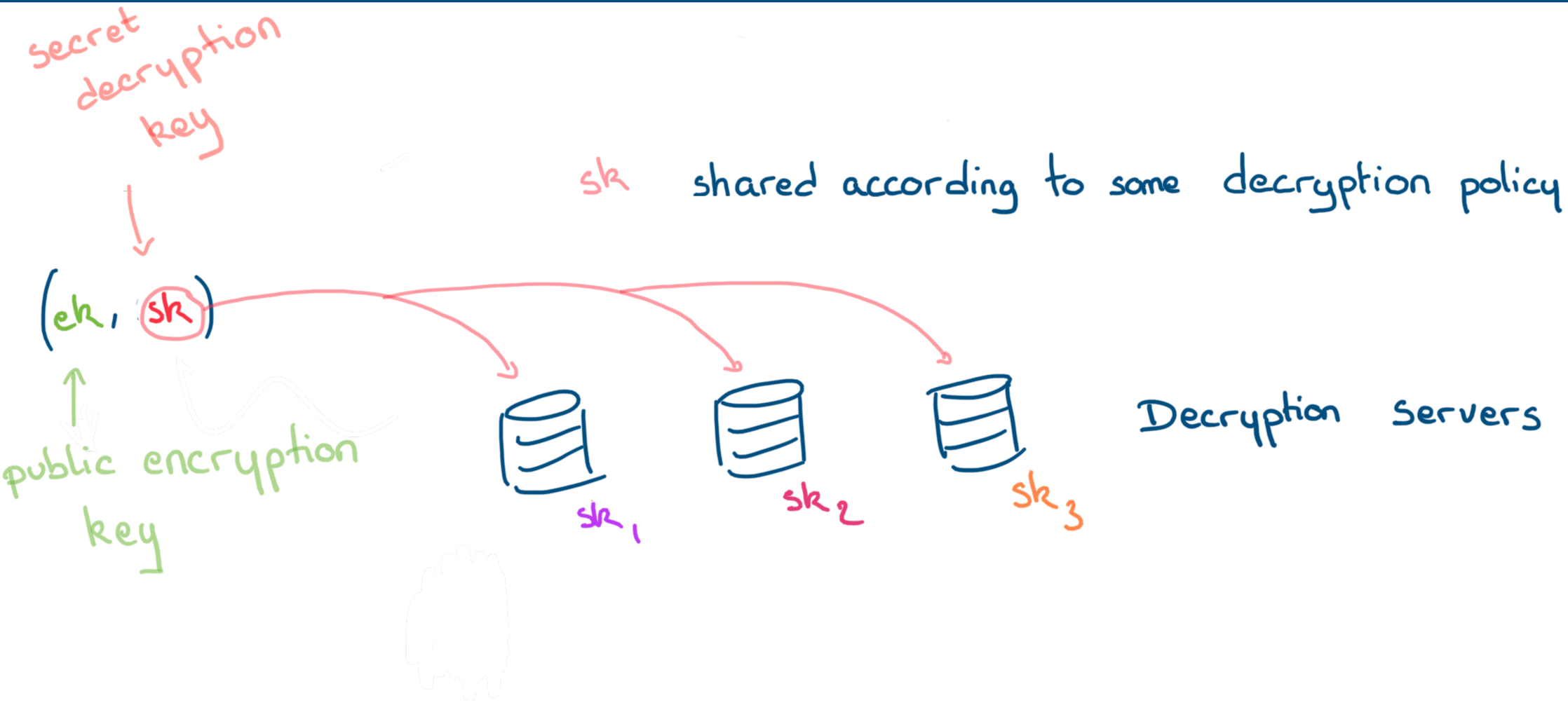
Guilhem Castagnos [1], Fabien Laguillaumie [2], **Ida Tucker** [3]

[1] Université de Bordeaux
[2] Université de Montpellier
[3] Zondax AG

Asiacrypt 2022

Threshold Public Key Encryption

Threshold Public Key Encryption

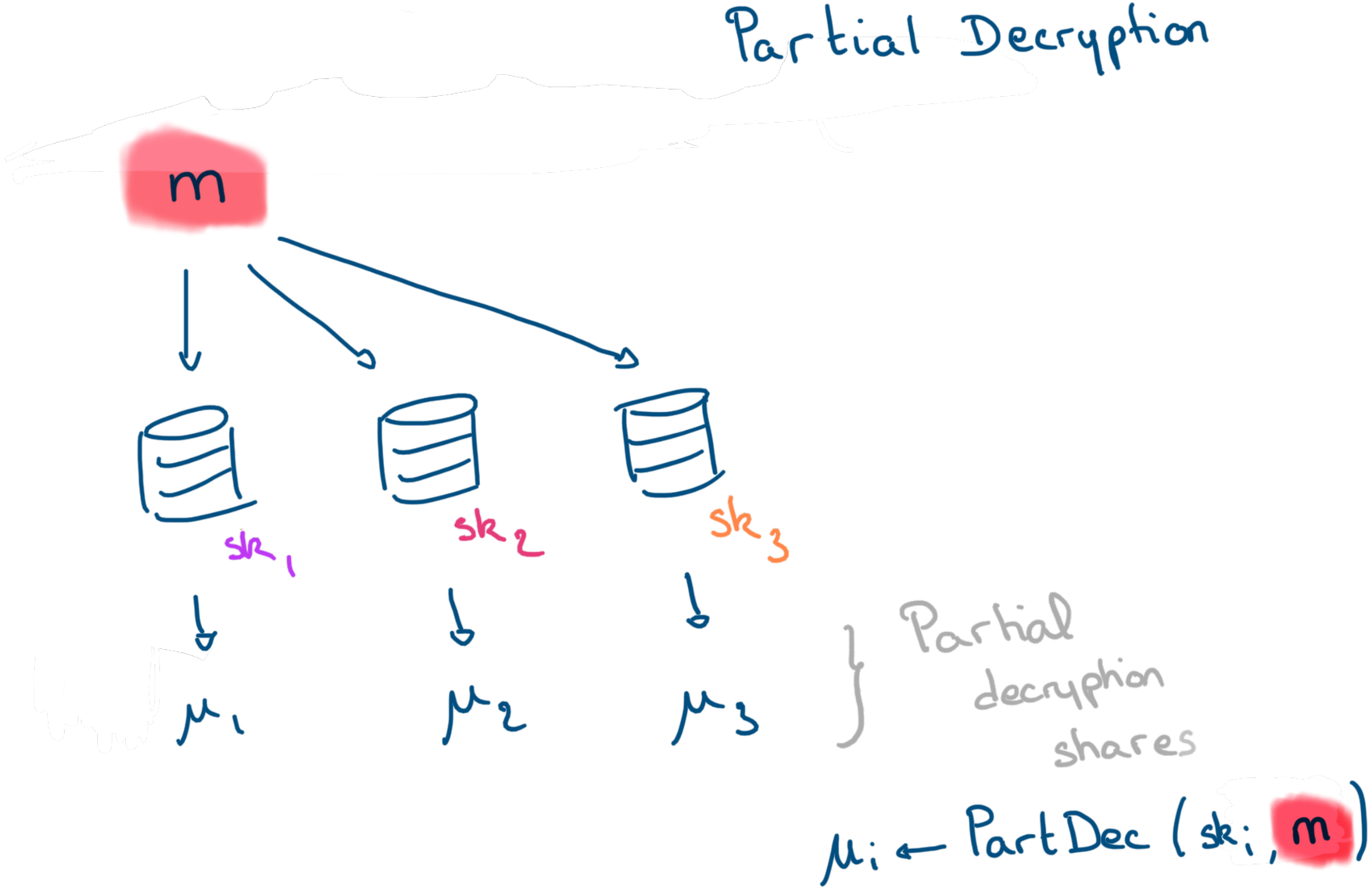


Threshold Public Key Encryption

Anyone can encrypt a message m with ek

m ← Encrypt(ek, m)

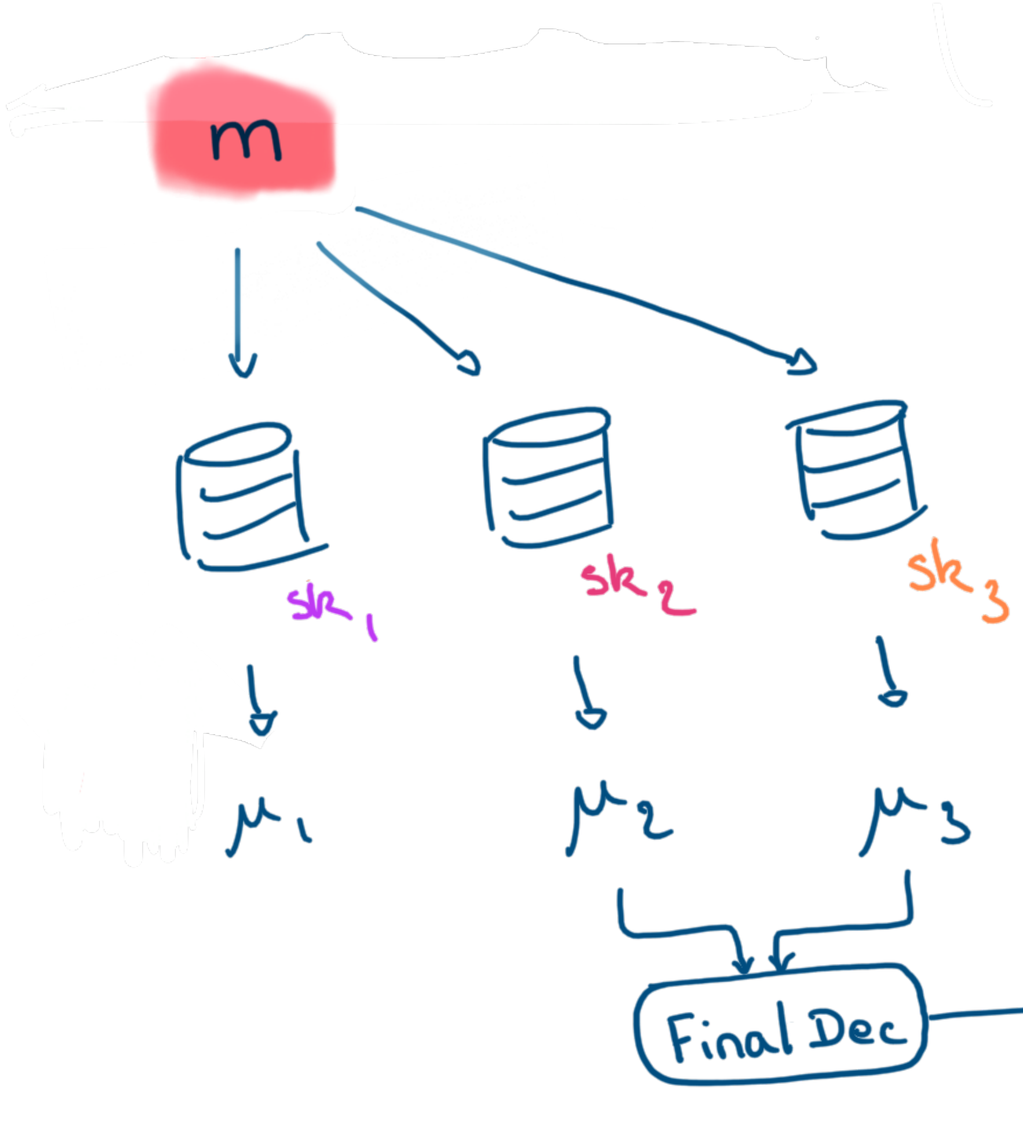
Threshold Public Key Encryption



Threshold Public Key Encryption

Final Decryption

public operation



$$\begin{cases} m & \text{if } \{2,3\} \text{ is a qualified set} \\ \perp & \text{otherwise.} \end{cases}$$

LINEARLY HOMOMORPHIC

Threshold Public Key Encryption

Given m that decrypts to m

and m^* that decrypts to m^*

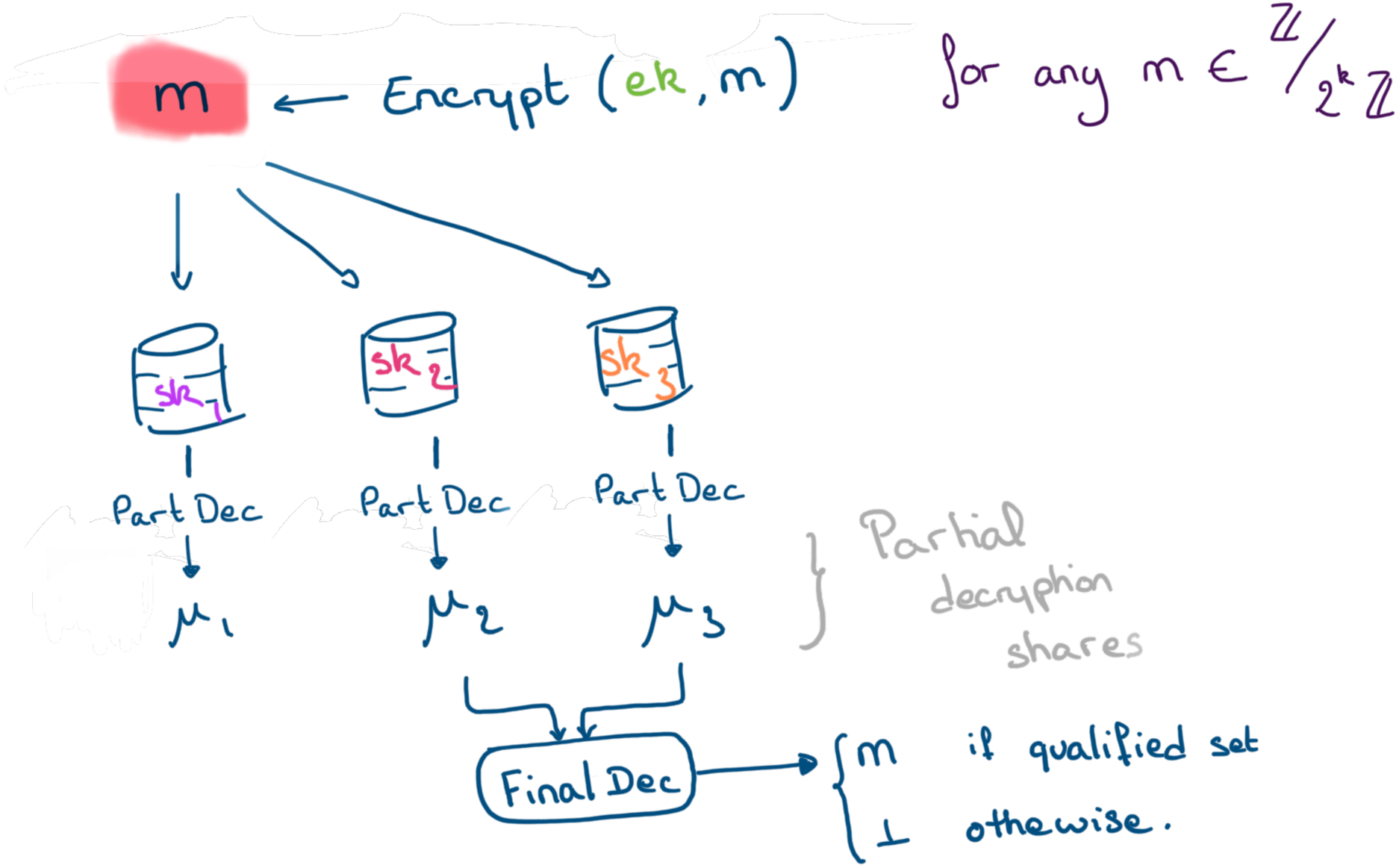
there exist efficient algorithms:

- $\text{Eval}_{\text{Add}}(m, m^*) \rightarrow m + m^*$
  - ↑ decrypts to $m + m^*$
- $\text{Eval}_{\text{Scal}}(m, \alpha) \rightarrow \alpha m^*$
  - ↑ decrypts to $\alpha \cdot m$
  - ↑ scalar

LINEARLY HOMOMORPHIC Threshold Public Key Encryption on $\mathbb{Z}/2^k\mathbb{Z}$

Public operations on ciphertexts

⇝ operations on underlying plaintexts modulo 2^k



Motivation

Multi-party computation modulo 2^k

Secure multi-party computation (MPC)

Alice

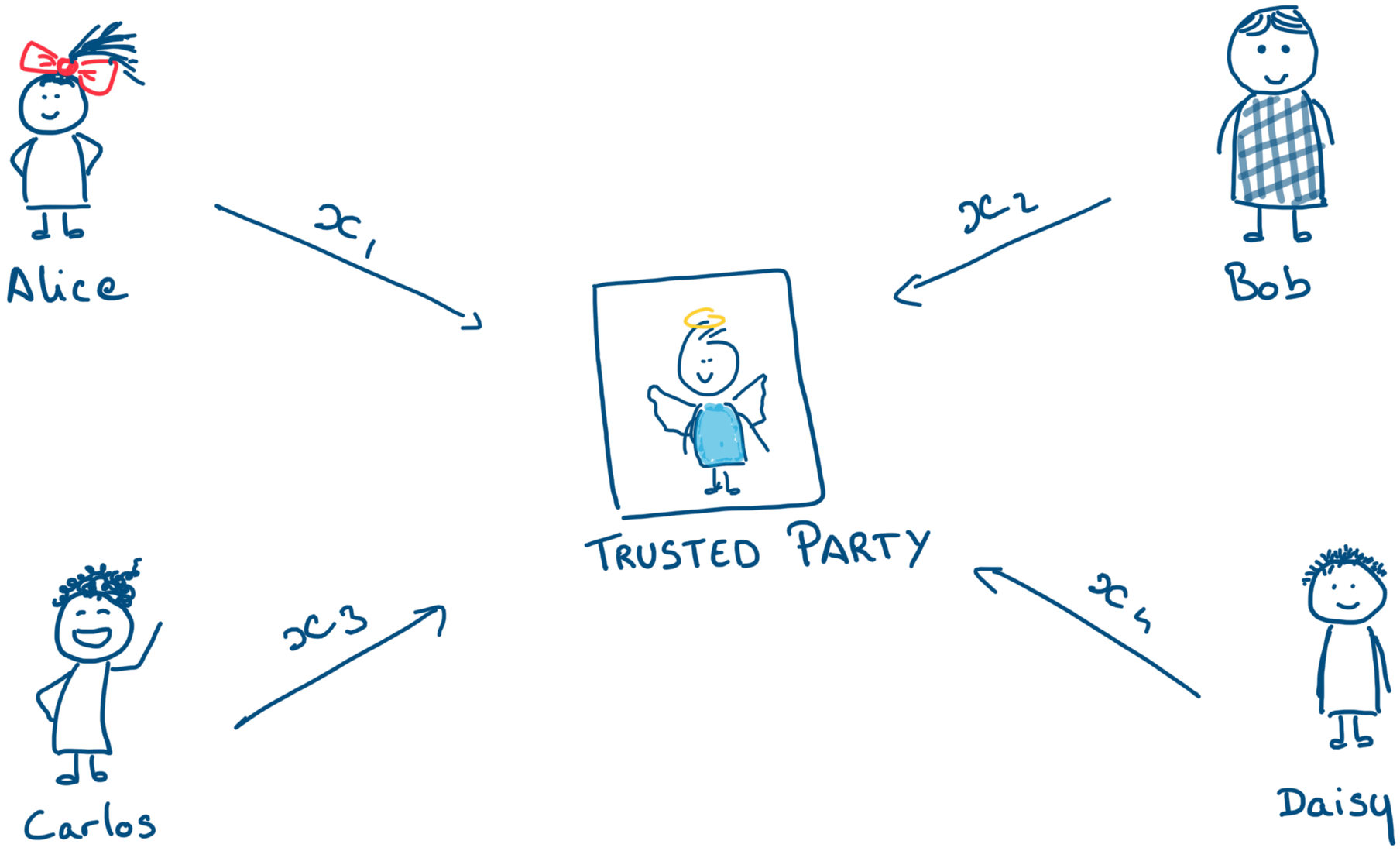
Bob

TRUSTED PARTY

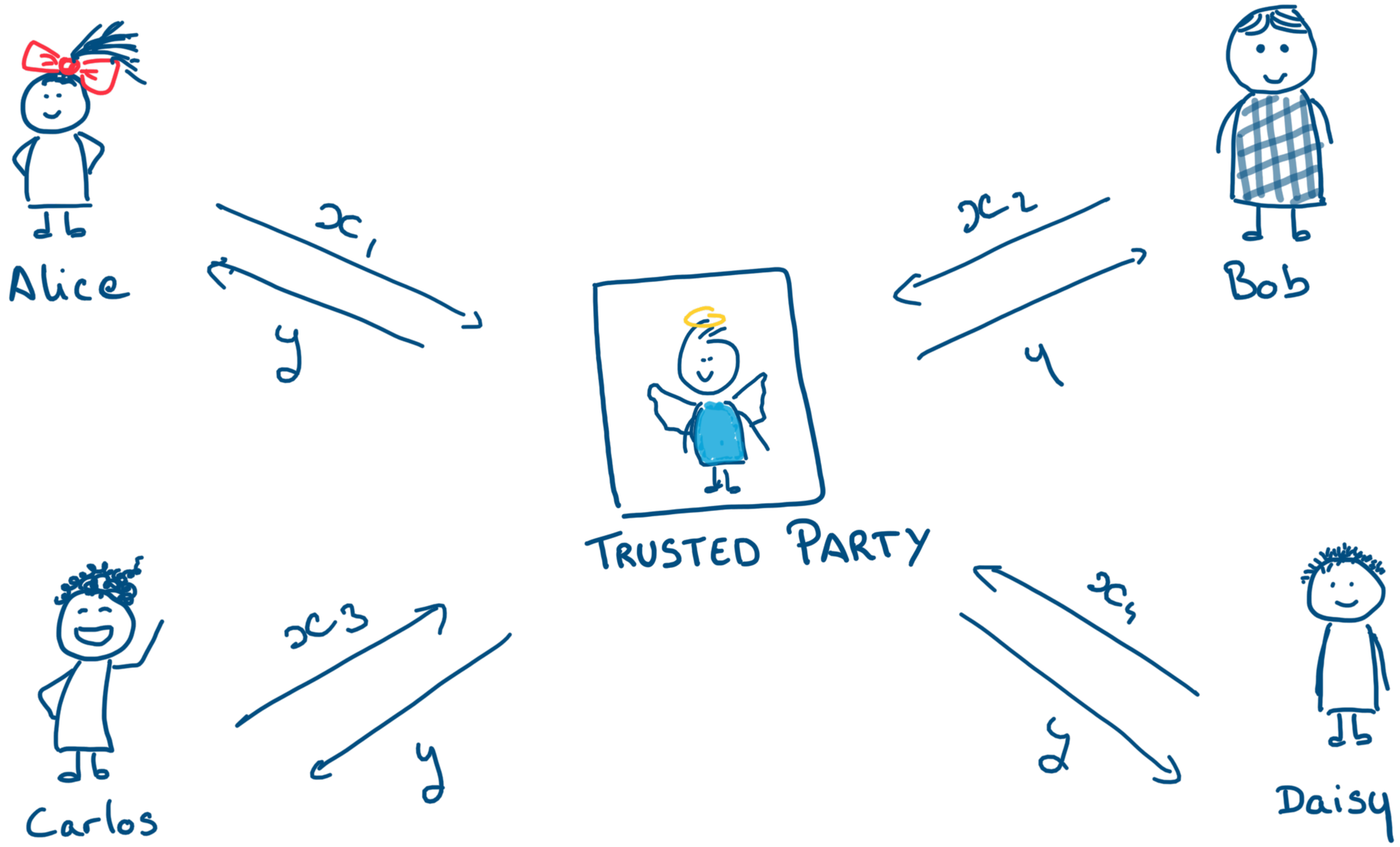
Carlos

Daisy

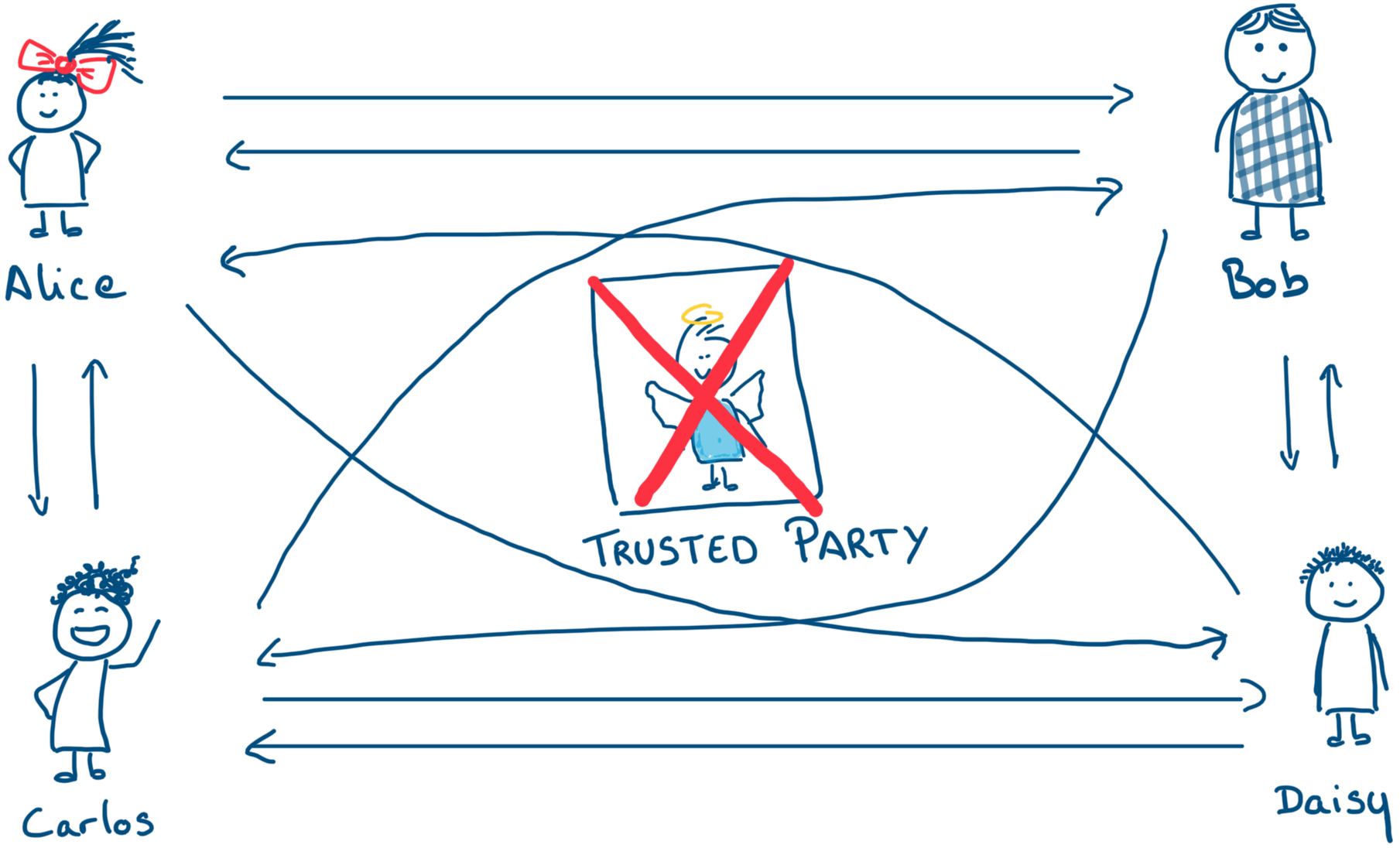
Secure multi-party computation (MPC)



Secure multi-party computation (MPC)

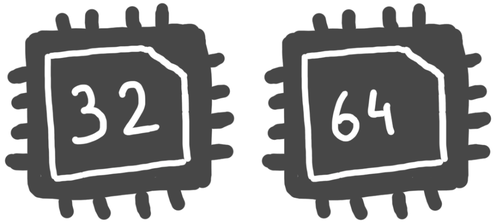


Secure multi-party computation (MPC)



Multi-party computation mod 2^k

Integer arithmetic on standard CPUs is done modulo 2^k (e.g. $32/64$ bits).



MPC modulo 2^k mirrors this design, and allows for:

- Simplified implementation

  No need for modular arithmetic, or to compensate modular reduction

- Using optimizations for CPUs directly

  Often expensive to emulate modulo p

Building multi-party computation mod 2^k

Preprocessing phase
(offline, slow)

Produces ‘raw material’ (a.k.a. correlated randomness)

Create shares of many triplets [a], [b], [ab],
where a and b are random in $\mathbb{Z}/2^k\mathbb{Z}$

Online phase
(fast, information theoretic)

Uses ‘raw material’ generated offline

Building multi-party computation mod 2^k

Preprocessing phase
(offline, slow)

Produces ‘raw material’ (a.k.a. correlated randomness)

Create shares of many triplets [a], [b], [ab],
where a and b are random in $\mathbb{Z}/2^k\mathbb{Z}$

- SPD$\mathbb{Z}_{2^k}$: Oblivious transfer (fast but high bandwidth consumption)
  Cramer, Damgård, Escudero, Scholl, and Xing @ CRYPTO 2018
- Overdrive2k : Somewhat homomorphic encryption (complex)
  Orsini, Smart, and Vercauteren @ CT-RSA 2020
- Mon$\mathbb{Z}_{2^k}$a : Linearly Homomorphic Encryption modulo 2^k
  Catalano, Di Raimondo, Fiore, and Giacomelli @ PKC 2020

Mon$\mathbb{Z}_{2^k}$a Catalano, Di Raimondo, Fiore, and Giacomelli @ PKC 2020

Uses the Joye-Libert encryption scheme

Linearly homomorphic

&

Message space $\mathbb{Z}/2^k\mathbb{Z}$

No known threshold decryption

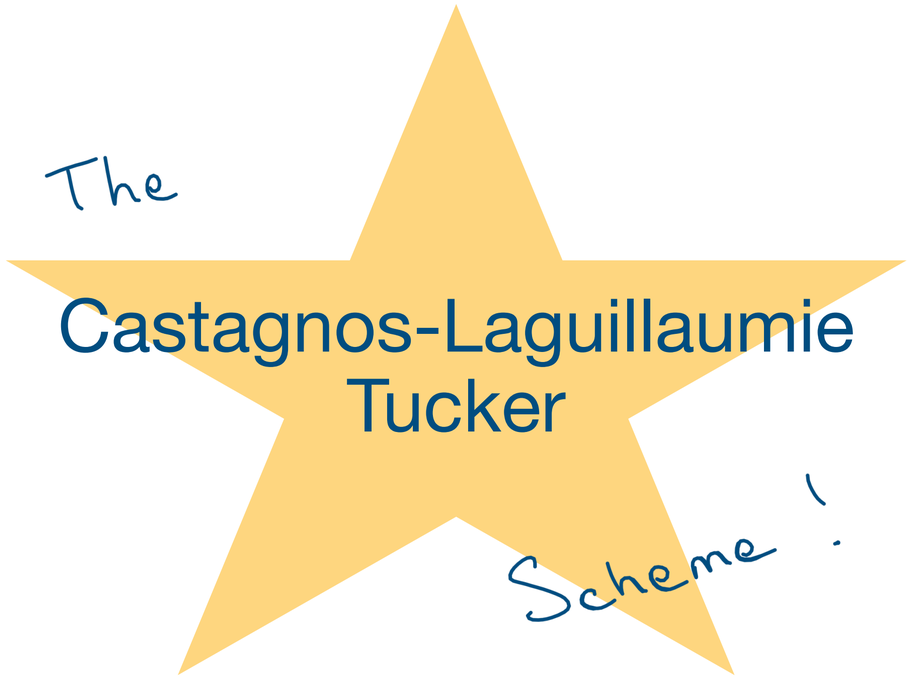
2-party MPC only
with
Zero Knowledge Proofs

Negative impact on bandwidth

Benhamouda, Herranz, Joye, and Libert @ JoC 2017

Our encryption scheme

Message space $\mathbb{Z}/2^k\mathbb{Z}$



Linearly homomorphic

Threshold decryption

Abstract framework
and construction

Abstract framework

Similar to the framework of Castagnos and Laguillaumie @ CT-RSA 2015

- $G = \langle g \rangle$: cyclic group of order $s \cdot 2^k$, with $gcd(2^k, s) = 1$
- $F = \langle f \rangle$: subgroup of 2^k–roots of unity of G, of order 2^k
- Efficient algorithm for computing discrete logarithms in F (x given f^x)
- $H = \langle h \rangle$: subgroup of 2^k–th powers of G of unknown odd order s
- $G \simeq F \times H$
- $\tilde{s}$ is a known upper bound for s

Abstract framework

Similar to the framework of Castagnos and Laguillaumie @ CT-RSA 2015

- $G = \langle g \rangle$: cyclic group of order $s \cdot 2^k$, with $gcd(2^k, s) = 1$
- $F = \langle f \rangle$: subgroup of 2^k–roots of unity of G, of order 2^k
- Efficient algorithm for computing discrete logarithms in F (x given f^x)
- $H = \langle h \rangle$: subgroup of 2^k–th powers of G of unknown odd order s
- $G \simeq F \times H$
- $\tilde{s}$ is a known upper bound for s

$z \hookleftarrow H \times F^x \approx_c z \hookleftarrow H$

HSM$_{2^k}$ assumption: Given $f, h, \tilde{s}$ and z_b , where $b \leftarrow \mathcal{U}\{0,1\}$,

$z_0 = h^x \cdot f^u$, and $z_1 = h^x$ for $x \hookleftarrow \mathcal{D}_H$ and $u \leftarrow \mathcal{U}((\mathbb{Z}/2^k\mathbb{Z})^\times)$

no PPT algorithm can decide b with probability significantly greater than 1/2.

Public Key Encryption Scheme on $\mathbb{Z}/2^k\mathbb{Z}$

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then
  return $\perp$
- return $\log_f(M)$

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Linearly Homomorphic Encryption Scheme on $\mathbb{Z}/2^k\mathbb{Z}$

KeyGen (pp):

- sample $sk \hookleftarrow \mathscr{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathscr{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then
  return $\perp$
- return $\log_f(M)$

EvalAdd ($pp, pk, (c_1, c_2), (c_1', c_2')$):

- $c_1'' := c_1\, c_1'$ and $c_2'' := c_2\, c_2'$
- sample $r \hookleftarrow \mathscr{D}_H$
- return $(c_1'' h^r,\ c_2'' pk^r)$

EvalScal ($pp, pk, (c_1, c_2), \alpha$):

- $c_1' := c_1^\alpha$ and $c_2' := c_2^\alpha$
- sample $r \hookleftarrow \mathscr{D}_H$
- return $(c_1' h^r,\ c_2' pk^r)$

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Security - game 0

Simulation

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_G$
- $pk := h^{sk}$
- return (pk, sk)

Distribution of public key in real and simulated KeyGen are negligibly close.

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then
  return $\perp$
- return $\log_f(M)$

Security - game 1

Simulation

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_G$
- $pk := h^{sk}$
- return (pk, sk)

Distribution of public key in real and simulated KeyGen are negligibly close.

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m c_1^{sk}$
- return (c_1, c_2)

Nothing changes,
just the way we compute c_2

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then
  return $\perp$
- return $\log_f(M)$

Security - game 2

Simulation

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_G$
- $pk := h^{sk}$
- return (pk, sk)

Distribution of public key in real and simulated KeyGen are negligibly close.

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$, $u \leftarrow (\mathbb{Z}/2^k\mathbb{Z})^\times$
- $c_1 := h^r f^u$
- $c_2 := f^m c_1^{sk}$
- return (c_1, c_2)

The HSM_{2^k} assumption ensures this change is indistinguishable

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then return $\perp$
- return $\log_f(M)$

Security - game 2

Simulation

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_G$
- $pk := h^{sk}$
- return (pk, sk)

Distribution of public key in real and simulated KeyGen are negligibly close.

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$, $u \leftarrow (\mathbb{Z}/2^k\mathbb{Z})$
- $c_1 := h^r f^u$
- $c_2 := f^m c_1^{sk}$
- return (c_1, c_2)

The HSM_{2^k} assumption ensures this change is indistinguishable

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then return $\perp$
- return $\log_f(M)$

Information on m in simulated cipher text:

$$c_2 = f^{m+u \cdot sk} pk^r$$

Security - game 2

Simulation

KeyGen (pp):

- sample $sk \hookleftarrow \mathscr{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

KeyGen (pp):

- sample $sk \hookleftarrow \mathscr{D}_G$
- $pk := h^{sk}$
- return (pk, sk)

Distribution of public key in real and simulated KeyGen are negligibly close.

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathscr{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathscr{D}_H$, $u \leftarrow (\mathbb{Z}/2^k\mathbb{Z})$
- $c_1 := h^r f^u$
- $c_2 := f^m c_1^{sk}$
- return (c_1, c_2)

The HSM_{2^k} assumption ensures this change is indistinguishable

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then return $\perp$
- return $\log_f(M)$

Information on m in simulated cipher text:

$$c_2 = f^{m+u \cdot sk} pk^r \longrightarrow m + u \cdot sk \mod 2^k$$

Information theoretically reveals

Security - game 2

Simulation

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_G$
- $pk := h^{sk}$
- return (pk, sk)

Distribution of public key in real and simulated KeyGen are negligibly close.

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$, $u \leftarrow (\mathbb{Z}/2^k\mathbb{Z})$
- $c_1 := h^r f^u$
- $c_2 := f^m c_1^{sk}$
- return (c_1, c_2)

The HSM_{2^k} assumption ensures this change is indistinguishable

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then return $\perp$
- return $\log_f(M)$

Information on m in simulated cipher text:

close to uniform mod 2^k

$$c_2 = f^{m+u \cdot sk} pk^r \longrightarrow m + u \cdot sk \bmod 2^k$$

Information theoretically reveals

Security - game 2

Simulation

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_G$
- $pk := h^{sk}$
- return (pk, sk)

Distribution of public key in real and simulated KeyGen are negligibly close.

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$, $u \leftarrow (\mathbb{Z}/2^k\mathbb{Z})$
- $c_1 := h^r f^u$
- $c_2 := f^m c_1^{sk}$
- return (c_1, c_2)

The HSM_{2^k} assumption ensures this change is indistinguishable

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then return $\perp$
- return $\log_f(M)$

Information on m in simulated cipher text:

$$c_2 = f^{m+u \cdot sk} pk^r \longrightarrow m + u \cdot sk \bmod 2^k$$

Invertible mod 2^k

close to uniform mod 2^k

Information theoretically reveals

Security - game 2

Simulation

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_H$
- $pk := h^{sk}$
- return (pk, sk)

KeyGen (pp):

- sample $sk \hookleftarrow \mathcal{D}_G$
- $pk := h^{sk}$
- return (pk, sk)

Distribution of public key in real and simulated KeyGen are negligibly close.

$G = \langle g \rangle$ of order $s \cdot 2^k$, $gcd(s, 2^k) = 1$

$F = \langle f \rangle$ of order 2^k

$H = \langle h \rangle$ of unknown order s

$G \simeq F \times H$

$\tilde{s}$ is a known upper bound for s

$pp := (f, h, \tilde{s})$ are public parameters

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$
- $c_1 := h^r$
- $c_2 := f^m pk^r$
- return (c_1, c_2)

Encrypt (pp, pk, m):

- sample $r \hookleftarrow \mathcal{D}_H$, $u \leftarrow (\mathbb{Z}/2^k\mathbb{Z})$
- $c_1 := h^r f^u$
- $c_2 := f^m c_1^{sk}$
- return (c_1, c_2)

The HSM_{2^k} assumption ensures this change is indistinguishable

Decrypt ($pp, sk, (c_1, c_2)$):

- $M := c_2 \cdot c_1^{-sk}$
- if $M \notin F$ then return $\perp$
- return $\log_f(M)$

Information on m in simulated cipher text:

$$c_2 = f^{m+u \cdot sk} pk^r$$

Invertible mod 2^k

close to uniform mod 2^k

$m + u \cdot sk \mod 2^k$

Information theoretically reveals

Perfectly masked!

Threshold variant

Decryption key sk is used for exponentiation to sk in group of **unknown** order H.

We use **linear integer secret sharing** (LISS) Damgård and Thorbek @ PKC 2006
to share the decryption key **over the integers.**

Our resulting scheme allows **any access structure** for the **decryption policy**

Realizing the framework

class groups of imaginary quadratic fields

Plug and play?

The original ‘CL framework’ of Castagnos, Laguillaumie @ CT-RSA 2015 provides a similar abstraction, only the subgroup F encoding messages is of **prime** order q.

Plug and play?

The original ‘CL framework’ of Castagnos, Laguillaumie @ CT-RSA 2015 provides a similar abstraction, only the subgroup F encoding messages is of **prime** order q.

Original CL with q prime

Two class groups:

- $Cl(\Delta_K)$ with discriminant $\Delta_K = -pq$
- $Cl(\Delta)$ with discriminant $-pq^3$.

Surjection $\bar{\phi}_q : Cl(\Delta) \twoheadrightarrow Cl(\Delta_K)$

$F = Ker(\bar{\phi}_q)$ of order q

Plug and play?

The original ‘CL framework’ of Castagnos, Laguillaumie @ CT-RSA 2015 provides a similar abstraction, only the subgroup F encoding messages is of **prime** order q.

Original CL with q prime

Plugging $q = 2^k$

Two class groups:
- $Cl(\Delta_K)$ with discriminant $\Delta_K = -pq$
- $Cl(\Delta)$ with discriminant $-pq^3$.

Surjection $\bar{\phi}_q : Cl(\Delta) \twoheadrightarrow Cl(\Delta_K)$

$F = Ker(\bar{\phi}_q)$ of order q

Known factorization of Δ_K

Can compute square roots, and decide if elements are squares.

HSM_{2^k} assumption does not hold!

Realizing the framework for $q = 2^k$

Computing square roots **must be hard** $\longrightarrow$ build Δ_K from **RSA integer** N

Realizing the framework for $q = 2^k$

Computing square roots **must be hard** ⟶ build Δ_K from **RSA integer** N

Genus theory associated to class groups:
Some genera can leak information on discrete logarithms!

Realizing the framework for $q = 2^k$

Computing square roots **must be hard** ⟶ build Δ_K from **RSA integer** N

Genus theory associated to class groups:
Some genera can leak information on discrete logarithms!

Carefully select discriminants Δ_K (and hence N) that allow
to securely work with the group of squares.

Performance

timings and cipher text size

BICYCL : C/C++ class group library

Bouvier, Castagnos, Imbert, Laguillaumie @ eprint.iacr.org/2022/1466

k	λ (bits)	Ciphertext (bits)	Setup	KeyGen	Encrypt	Decrypt
64	112	3272	0.571 s	0.019 s	7.78 ms	17.7 ms
	128	4808	1.78 s	0.044 s	17.4 ms	40.1 ms

N has sizes of 2048 bits (for λ = 112) and 3072 bits (λ = 128).

Implemented from scratch in C++.
Timings performed on a standard laptop (Intel(R) Core(TM) i7-8665U CPU @ 1.90GHz).

That's all folks

Questions?