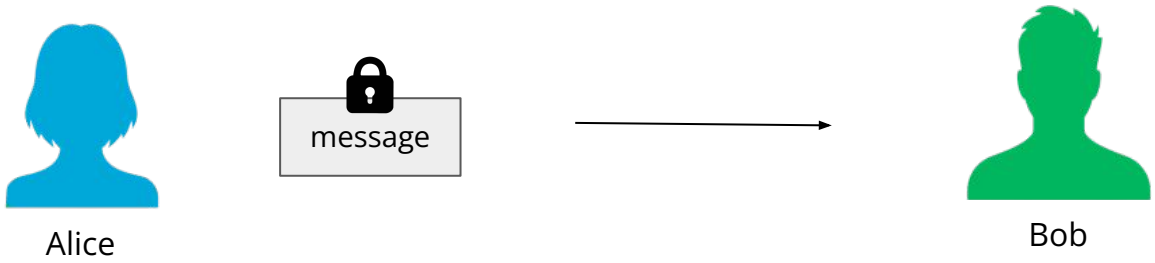
Inner-Product Predicate Encryption from Weaker Assumptions

Maya Kalai and Ella Kim
Mentor: Sacha Servan-Schreiber

MIT PRIMES October Conference
October 12, 2024

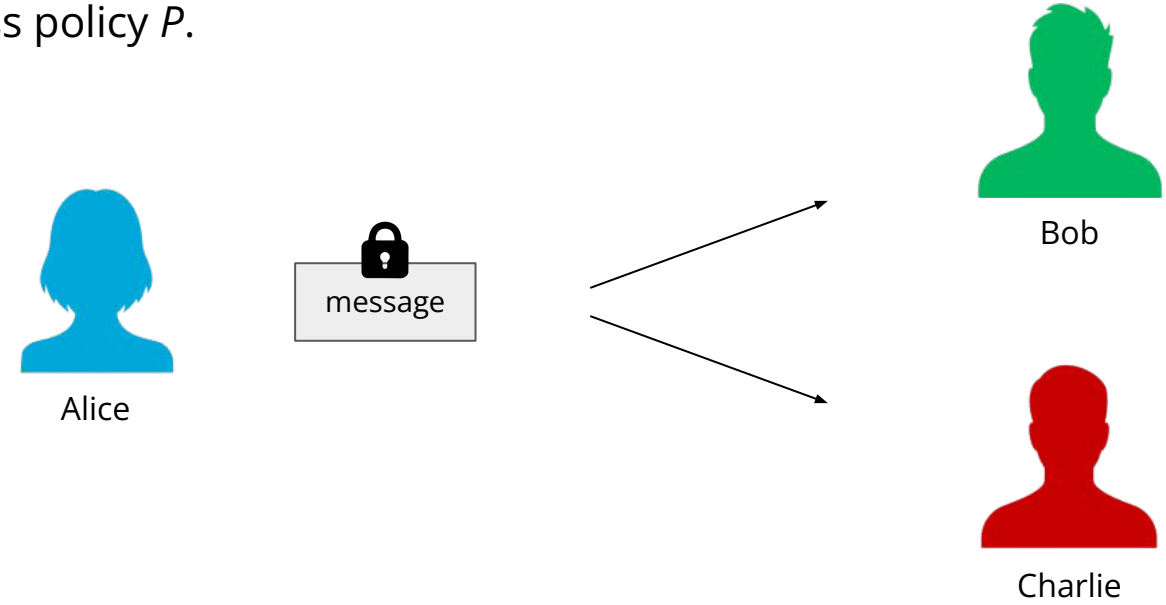
Encryption

Alice wants to encrypt a message and send it to Bob.



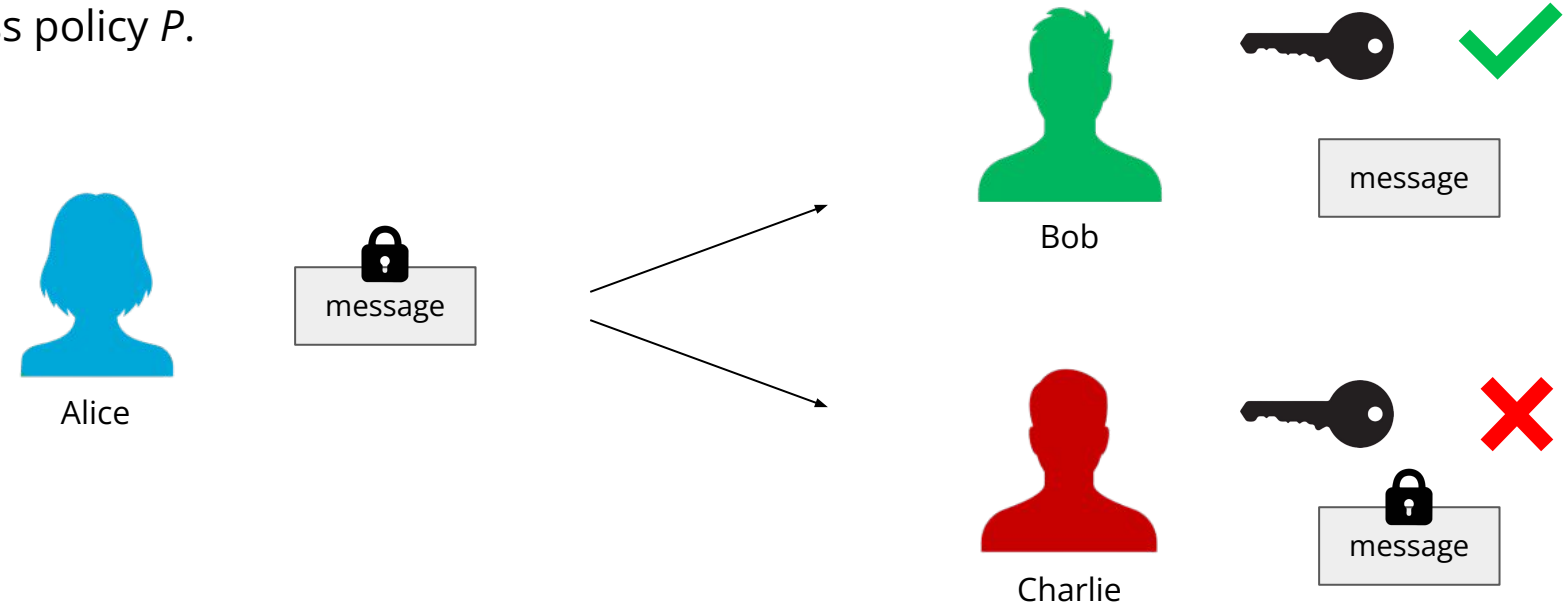
Predicate Encryption

Alice wants to encrypt a message and send it to Bob and Charlie. However, she wants them to be able to decrypt the message if and only if they have a key that satisfies an access policy *P*.



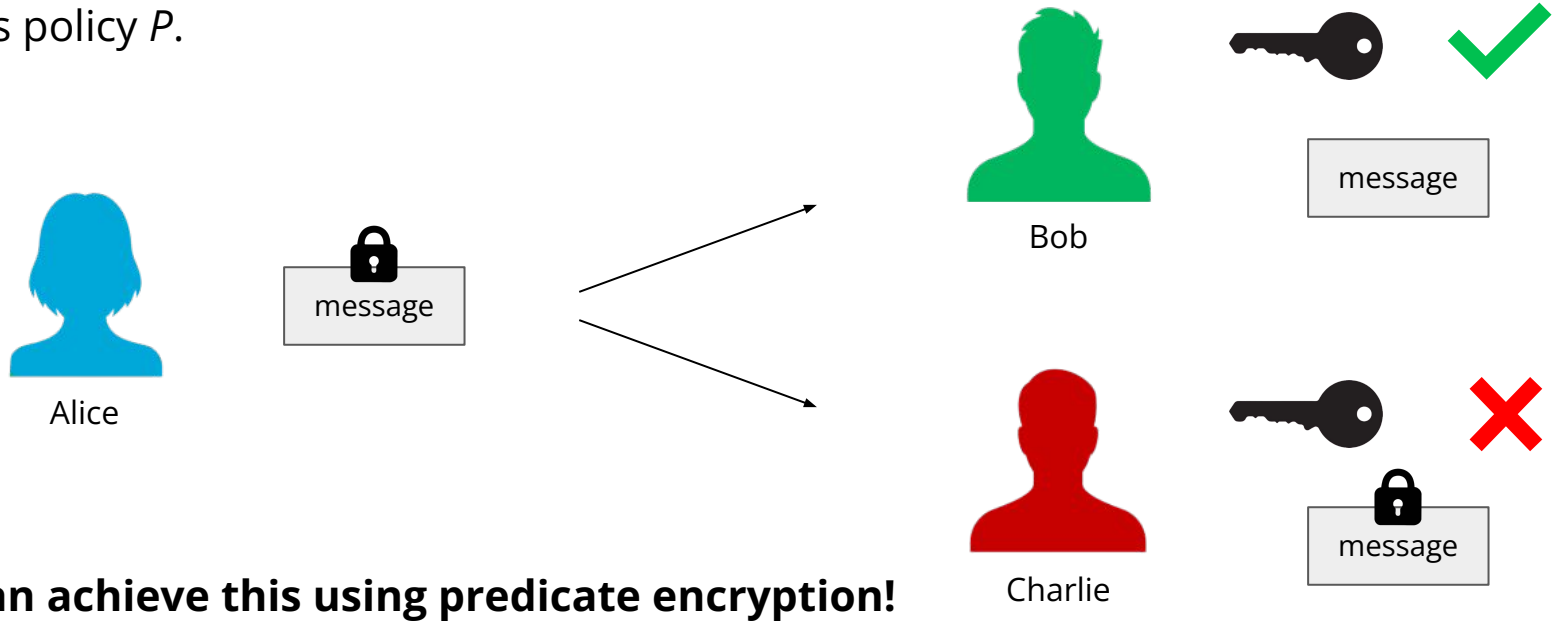
Predicate Encryption

Alice wants to encrypt a message and send it to Bob and Charlie. However, she wants them to be able to decrypt the message if and only if they have a key that satisfies an access policy *P*.



Predicate Encryption

Alice wants to encrypt a message and send it to Bob and Charlie. However, she wants them to be able to decrypt the message if and only if they have a key that satisfies an access policy *P*.



We can achieve this using predicate encryption!

Encryption

A secret-key encryption scheme has three algorithms.

- KeyGen → secret key *sk*
- Enc(*sk*, message *m*) → ciphertext *ct*
- Dec(*sk*, *ct*) → *m*

  Correctness: Dec(*sk*, Enc(*sk*, *m*)) = *m*

  Security: Enc(*sk*, *m*) “hides” *m*

Predicate Encryption

A predicate encryption scheme has four algorithms and a predicate class P.

- Setup → master key *msk*
- Enc(*msk*, attribute *x,* message *m*) → ciphertext *ct*
- KeyGen(*msk*, *P*) → secret key sk_P
- Dec(sk_P, *ct*) → *m*

Correctness: Dec(sk_P, Enc(*msk*, *m*)) = *m* if and only if *P*(*x*) = 0.

Security: Enc(*msk*, *x*, *m*) “hides” *m* even if the adversary has sk_P where *P*(*x*) ≠ 0.

Example - Credit Card Transactions

plaintext *m*: Name

attribute **x**: cost

	Name	Cost	Time	Zip Code
🔒	Alice	$25	9/25/24 15:34	02139
🔒	Alice	$3	9/28/24 18:23	02139
🔒	Alice	$3250	10/3/24 10:11	02139
🔒	Alice	$200	10/5/24 16:45	02140
🔒	Alice	$1200	10/6/24 09:31	02139
	...			
🔒	Bob	$5	9/27/24 18:23	02139
🔒	Bob	$1700	9/30/24 10:49	02139
🔒	Bob	$200	9/30/24 17:45	02140
🔒	Bob	$2	10/5/24 18:31	02140
🔒	Bob	$2300	10/7/24 13:12	02139
🔒	Bob	$1500	10/8/24 07:03	02139

Example - Credit Card Transactions

Fraud Investigation

- Cost: >$1000

	Name	Cost	Time	Zip Code
🔒	Alice	$25	9/25/24 15:34	02139
🔒	Alice	$3	9/28/24 18:23	02139
🔒	Alice	$3250	10/3/24 10:11	02139
🔒	Alice	$200	10/5/24 16:45	02140
🔒	Alice	$1200	10/6/24 09:31	02139
	...			
🔒	Bob	$5	9/27/24 18:23	02139
🔒	Bob	$1700	9/30/24 10:49	02139
🔒	Bob	$200	9/30/24 17:45	02140
🔒	Bob	$2	10/5/24 18:31	02140
🔒	Bob	$2300	10/7/24 13:12	02139
🔒	Bob	$1500	10/8/24 07:03	02139

Example - Credit Card Transactions

Predicate *P*: "Is the cost greater than $1000?"

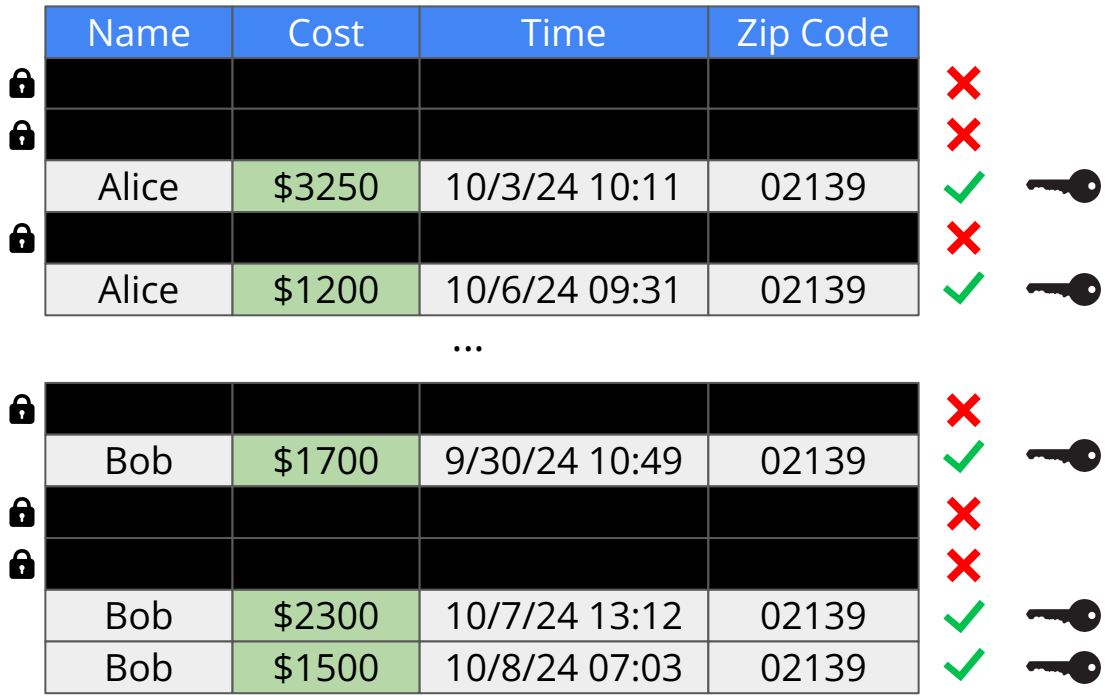
If yes, we can give a secret key sk_P to decrypt the name.

Name	Cost	Time	Zip Code
Alice	$25	9/25/24 15:34	02139
Alice	$3	9/28/24 18:23	02139
Alice	$3250	10/3/24 10:11	02139
Alice	$200	10/5/24 16:45	02140
Alice	$1200	10/6/24 09:31	02139
...			
Bob	$5	9/27/24 18:23	02139
Bob	$1700	9/30/24 10:49	02139
Bob	$200	9/30/24 17:45	02140
Bob	$2	10/5/24 18:31	02140
Bob	$2300	10/7/24 13:12	02139
Bob	$1500	10/8/24 07:03	02139

Example - Credit Card Transactions

Predicate *P*: “Is the cost greater than $1000?”

If yes, we can give a secret key sk_P to decrypt the name.



Inner Product Predicate Encryption (IPPE)

A inner product predicate encryption scheme has four algorithms and a predicate P. For a plaintext message m and attribute vector $\mathbf{x}$:

- Setup $\rightarrow$ master key msk
- Enc(msk, $\mathbf{x}$, m) $\rightarrow$ ciphertext ct
- KeyGen(msk, P) $\rightarrow$ secret key sk_P
- Dec(sk_P, ct) $\rightarrow$ m

We let the predicate be $P(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$, for a fixed predicate vector $\mathbf{y}$.

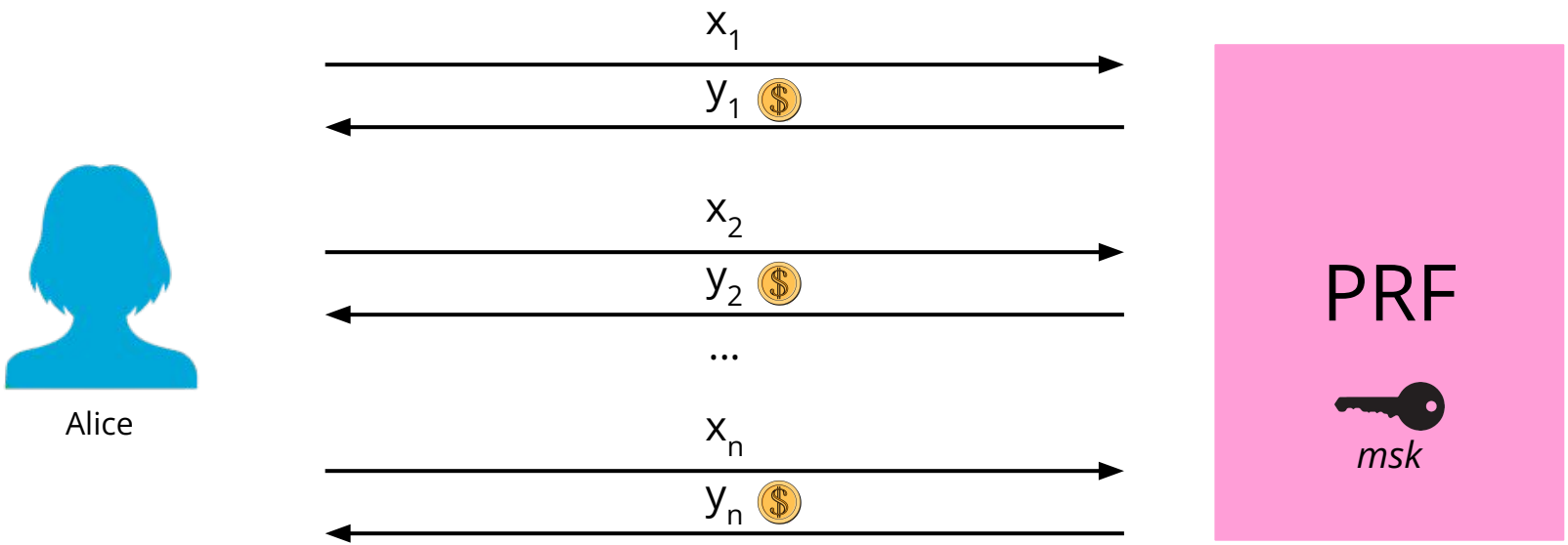
Correctness: Dec(sk_P, Enc(msk, m)) = m if and only if $P(\mathbf{x}) = 0$.

Security: Enc(msk, m) “hides” m even if the adversary has sk_P where $P(\mathbf{x}) \neq 0$.

Construction

Pseudorandom Function (PRF)

A pseudorandom function (PRF) is a function that returns a value computationally indistinguishable from random.



Constrained Pseudorandom Function (CPRF)

Alice

csk

x_1

y_1 ✓

x_2

y_2 ✓

...

x_n

y_n ✗

CPRF

msk

Constrained Pseudorandom Function (CPRF)

A constrained PRF is associated with 4 algorithms, a master key msk_{cprf}, constrained key csk, and constraint function C. We let $C = \langle z, x \rangle$ for a constraint vector z and input vector x.

- CPRF.KeyGen $\rightarrow msk_{cprf}$
- CPRF.Eval(msk_{cprf}, x) $\rightarrow y$
- CPRF.Constrain(msk_{cprf}, C) $\rightarrow csk$
- CPRF.CEval(csk, x) $\rightarrow y$ if $C(x) = 0$ and $\perp$ otherwise

Our scheme uses inner product constraints.

Shiftable Constrained Pseudorandom Function (ShCPRF)

A shiftable CPRF is associated with 4 algorithms, a master key msk_{shcprf}, constrained key *csk*, a constraint function *C,* and a shift *r*. We let $C = \langle z, x \rangle$ for a constraint vector *z* and input vector *x*.

- CPRF.KeyGen $\rightarrow msk_{\text{shcprf}}$
- CPRF.Eval(msk_{shcprf}, *x*) $\rightarrow y$
- CPRF.Constrain(msk_{shcprf}, *C*, *r*) $\rightarrow$ *csk*
- CPRF.CEval(*csk*, *x*) $\rightarrow y$ if $C(x) = r$ and random otherwise

Our scheme uses shiftable inner product constraints.

Main Idea - Using ShCPRFs to construct IPPE

Encryption (for message *m* in {0,1})

Generate ciphertext *ct*

Compute tag *t* = ShCPRF(*ct*, attribute **x**)

Output (*ct*, *x*, *t*)

Key Generation (corresponding to P(x)=⟨z,x⟩)

Generate secret key *csk* for ShCPRF

Decryption

Compute tag *t'* from ShCPRF.CEval using c*sk* and (ct, **x**)

Output *m* = 0 if and only if *t'* = *t*

Outline of Scheme

Alice

Bob

Setup(1^λ) $\rightarrow$ msk
Enc(msk, $\mathbf{x}$, m) $\rightarrow$ $ct_{\mathbf{x}}$
KeyGen(msk, $\mathbf{y}$) $\rightarrow$ $sk_{\mathbf{y}}$

$sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$

Dec($sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$, $\mathbf{y}$) $\rightarrow$ m or $\perp$

Overview of Scheme

$P(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$ for attribute vector $\mathbf{x}$ and predicate vector $\mathbf{y}$

Setup$(1^\lambda) \rightarrow msk$. The setup algorithm gets the security parameter 1^λ as input and outputs the master key msk.

Setup$(1^\lambda) \rightarrow msk$
Enc$(msk, \mathbf{x}, m) \rightarrow ct_{\mathbf{x}}$
KeyGen$(msk, \mathbf{y}) \rightarrow sk_{\mathbf{y}}$

$sk_{\mathbf{y}}, ct_{\mathbf{x}}$

Dec$(sk_{\mathbf{y}}, ct_{\mathbf{x}}, \mathbf{y}) \rightarrow m$ or $\perp$

Setup

Setup(1^λ) $\rightarrow$ *msk*

- Generate $msk_{\text{shcprf}} \leftarrow$ ShCPRF.KeyGen(1^λ)
- Output $msk := msk_{\text{shcprf}}$

Overview of Scheme

$P(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$

Enc(msk, $\mathbf{x}$, m) $\rightarrow$ $ct_{\mathbf{x}}$. The encryption algorithm gets msk, the attribute $\mathbf{x}$, and message $m \in \{0, 1\}$ as input and outputs the ciphertext $ct_{\mathbf{x}}$.

Setup(1^{λ}) $\rightarrow$ msk

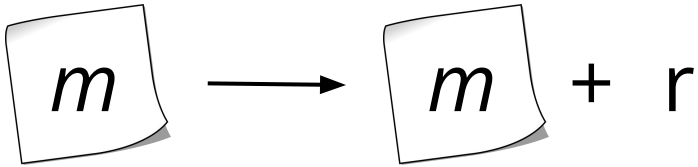
Enc(msk, $\mathbf{x}$, m) $\rightarrow$ $ct_{\mathbf{x}}$

KeyGen(msk, $\mathbf{y}$) $\rightarrow$ $sk_{\mathbf{y}}$

$sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$

Dec($sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$, $\mathbf{y}$) $\rightarrow$ m or $\perp$

Enc



Enc(msk, $\mathbf{x}$, m) $\rightarrow$ $ct_{\mathbf{x}}$

- Parse $msk = msk_{\text{shcprf}}$
- Sample $r \leftarrow \mathbf{F}_q$
- $ct := m + r$
- $t \leftarrow$ ShCPRF.Eval(msk_{shcprf}, (ct, $\mathbf{x}$), r)
- Output $ct_{\mathbf{x}} := (ct, \mathbf{x}, t)$

Overview of Scheme

$P(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$

KeyGen(msk, $\mathbf{y}$) $\rightarrow sk_{\mathbf{y}}$. The key generation algorithm gets msk and the predicate vector $\mathbf{y}$ as input and outputs the secret key $sk_{\mathbf{y}}$.

Setup(1^{λ}) $\rightarrow msk$
Enc(msk, $\mathbf{x}$, m) $\rightarrow ct_{\mathbf{x}}$
KeyGen(msk, $\mathbf{y}$) $\rightarrow sk_{\mathbf{y}}$

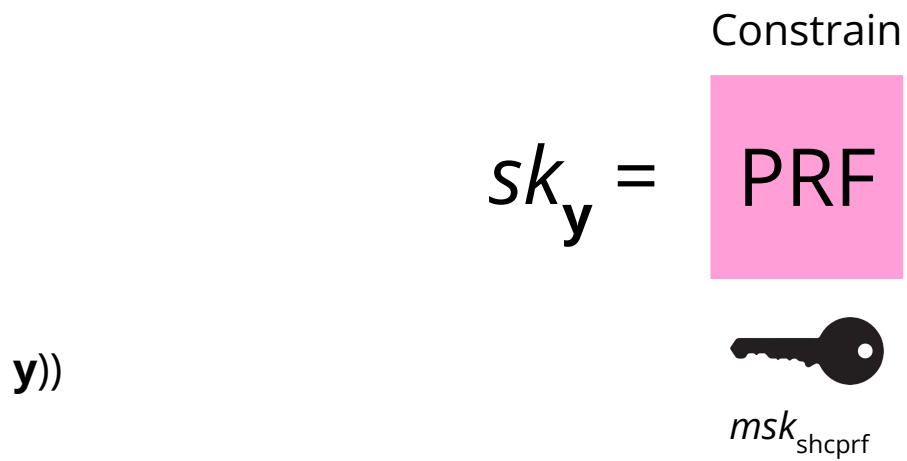
$sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$

Dec($sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$, $\mathbf{y}$) $\rightarrow m$ or $\perp$

KeyGen

KeyGen(msk, $\mathbf{y}$) $\rightarrow$ $sk_{\mathbf{y}}$

- Parse $msk = msk_{\text{shcprf}}$
- $sk_{\mathbf{y}} \leftarrow$ ShCPRF.Constrain(msk_{shcprf}, (1, $\mathbf{y}$))
- Output $sk_{\mathbf{y}}$



Overview of Scheme

$P(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$

$\mathrm{Dec}(sk_{\mathbf{y}}, ct_{\mathbf{x}}, \mathbf{y}) \rightarrow m$ or $\perp$. The decryption algorithm gets $sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$, $\mathbf{y}$ as input. It outputs the message m if $P(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle = 0$ and $\perp$ otherwise.

$\mathrm{Setup}(1^{\lambda}) \rightarrow msk$
$\mathrm{Enc}(msk, \mathbf{x}, m) \rightarrow ct_{\mathbf{x}}$
$\mathrm{KeyGen}(msk, \mathbf{y}) \rightarrow sk_{\mathbf{y}}$

$sk_{\mathbf{y}}, ct_{\mathbf{x}}$

$\mathrm{Dec}(sk_{\mathbf{y}}, ct_{\mathbf{x}}, \mathbf{y}) \rightarrow m$ or $\perp$

Dec

Dec($sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$, $\mathbf{y}$) $\rightarrow m$ or $\perp$

- Parse $ct_{\mathbf{x}} = (ct, \mathbf{x}, t)$
- $t' \leftarrow$ ShCPRF.CEval($sk_{\mathbf{y}}$, (ct, $\mathbf{x}$))

Dec

Dec($sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$, $\mathbf{y}$) $\rightarrow m$ or $\perp$

- Parse $ct_{\mathbf{x}} = (ct, \mathbf{x}, t)$
- $t' \leftarrow$ ShCPRF.CEval($sk_{\mathbf{y}}$, $(ct, \mathbf{x})$)

Reminders:

- shift r
- constraint vector $(1, \mathbf{y})$
- input vector $(ct, \mathbf{x})$

We want $\langle (1, \mathbf{y}), (ct, \mathbf{x}) \rangle = r$.

Dec

Dec($sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$, $\mathbf{y}$) $\rightarrow m$ or $\perp$

- Parse $ct_{\mathbf{x}} = (ct, \mathbf{x}, t)$
- $t' \leftarrow$ ShCPRF.CEval($sk_{\mathbf{y}}$, $(ct, \mathbf{x})$)

We want $m + r + \langle \mathbf{x}, \mathbf{y} \rangle = r$.

Reminders:

We want $\langle (1, \mathbf{y}), (ct, \mathbf{x}) \rangle = r$.

$ct = m + r$

$\langle (1, \mathbf{y}), (ct, \mathbf{x}) \rangle = ct + \langle \mathbf{x}, \mathbf{y} \rangle$

$= m + r + \langle \mathbf{x}, \mathbf{y} \rangle$

Dec

Dec($sk_{\mathbf{y}}$, $ct_{\mathbf{x}}$, $\mathbf{y}$) $\rightarrow m$ or $\perp$

Parse $ct_{\mathbf{x}} = (ct, \mathbf{x}, t)$

$t' \leftarrow$ ShCPRF.CEval($sk_{\mathbf{y}}$, $(ct, \mathbf{x})$)

We want $m + r + \langle \mathbf{x}, \mathbf{y} \rangle = r$.

If $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$, output $\perp$.

Given $\langle \mathbf{x}, \mathbf{y} \rangle = 0$,

- If $t = t'$, $m = 0$.
- If $t \neq t'$, $m = 1$.

This allows for us to recover m!

Charlie

Since Charlie has $\langle \mathbf{x}, \mathbf{y} \rangle \neq 0$, he cannot decrypt because t' will not be equal to t.

Bob

Since Bob has $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, he will check if $t' = t$. If it does, then $m = 0$, otherwise $m = 1$.

Current and Future Work

- Working on the proof of security
- Extending the scheme to support more features
- Implementing and benchmarking the construction

Acknowledgements

Thank you to

- Sacha Servan-Schreiber for mentoring our project.
- the MIT PRIMES organizers for providing this research opportunity.

References

Geoffroy Couteau, Lalita Devadas, Srinivas Devadas, Alexander Koch, and Sacha Servan-Schreiber. Quietot: Lightweight oblivious transfer with a public-key setup. *Cryptology ePrint Archive*, 2024.

Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Annual Cryptology Conference*, pages 503–523. Springer, 2015.

Sacha Servan-Schreiber. Constrained pseudorandom functions for inner-product predicates from weaker assumptions. *Cryptology ePrint Archive*, 2024.