An Introduction to Distributed Symmetric Encryption

Erik Pohle
erik.pohle@esat.kuleuven.be

April 1st, 2025

COSIC, KU Leuven, Belgium

COSIC

KU LEUVEN

Content

- Setting
- Applications
- DiSE
- MPC-based Distributed Encryption
- for AES

Symmetric-key Encryption

- Alice and Bob share a secret key k
- Alice computes $c \leftarrow \mathrm{Enc}(k, m)$
- Alice sends c to Bob
- Bob computes $m \leftarrow \mathrm{Dec}(k, c)$

Symmetric-key Encryption

- Alice and Bob share a secret key k
- Alice computes $c \leftarrow \mathrm{Enc}(k, m)$
- Alice sends c to Bob
- Bob computes $m \leftarrow \mathrm{Dec}(k, c)$

Security?

- The adversary does not know k
- Cannot learn anything about m given c
- IND-CPA, IND-CCA

Symmetric-key Encryption

- Alice and Bob share a secret key k
- Alice computes $c \leftarrow \mathrm{Enc}(k, m)$
- Alice sends c to Bob
- Bob computes $m \leftarrow \mathrm{Dec}(k, c)$

Security?

- The adversary does not know k
- Cannot learn anything about m given c
- IND-CPA, IND-CCA

Examples

AES-CBC, AES-GCM, ChaCha20-Poly1305, ...

Setting: Distributed Encryption

- Goal: distribute Alice/Bob's operations among $P_1, \ldots, P_n$
- Why? no small group of parties should learn the secret key k (threshold from secret sharing)
- P_i only knows $[\![k]\!]_i$

Setting: Distributed Encryption

- Goal: distribute Alice/Bob's operations among $P_1, \ldots, P_n$
- Why? no small group of parties should learn the secret key k (threshold from secret sharing)
- P_i only knows $[\![k]\!]_i$

Protocol Π_1

P_1 knows $m, [\![k]\!]_1$
P_i knows $[\![k]\!]_i$, $i \neq 1$

1. Parties run distributed encryption protocol
2. P_1 outputs c

Setting: Distributed Encryption

- Goal: distribute Alice/Bob's operations among $P_1, \ldots, P_n$
- Why? no small group of parties should learn the secret key k (threshold from secret sharing)
- P_i only knows $[\![k]\!]_i$

Protocol Π_1

P_1 knows $m, [\![k]\!]_1$
P_i knows $[\![k]\!]_i$, $i \neq 1$

1. Parties run distributed encryption protocol
2. P_1 outputs c

Protocol Π_2

P_i knows $[\![k]\!]_i, [\![m]\!]_i$

1. Parties run distributed encryption protocol
2. P_i outputs c

Distributed Decryption

Protocol Π_1

P_1 knows $c, [\![k]\!]_1$
P_i knows $[\![k]\!]_i$, $i \neq 1$

1. Parties run distributed decryption protocol
2. P_1 outputs m

Distributed Decryption

Protocol Π_1

P_1 knows $c, [\![k]\!]_1$
P_i knows $[\![k]\!]_i$, $i \neq 1$

1. Parties run distributed decryption protocol
2. P_1 outputs m

Protocol Π_2

P_i knows $[\![k]\!]_i, c$

1. Parties run distributed decryption protocol
2. P_i outputs $[\![m]\!]_i$

Distributed Decryption

Protocol Π_1

P_1 knows $c, [\![k]\!]_1$
P_i knows $[\![k]\!]_i$, $i \neq 1$

1. Parties run distributed decryption protocol
2. P_1 outputs m

Protocol Π_1 = DiSE

Protocol Π_2

P_i knows $[\![k]\!]_i, c$

1. Parties run distributed decryption protocol
2. P_i outputs $[\![m]\!]_i$

Π_2 = MPC-based distributed encryption/decryption

Desirable Properties

Desirable Properties

Correctness

$c = \mathsf{Enc}(k, m)$

Desirable Properties

Correctness

$c = \mathsf{Enc}(k, m)$

Privacy

- Corrupted P_j cannot learn anything about k
- (Π_1) If P_1 is honest, P_j cannot learn anything about m (except for its length)
- (Π_2) P_j cannot learn anything about m (except for its length)

Desirable Properties

Correctness

$c = \mathsf{Enc}(k, m)$

Privacy

- Corrupted P_j cannot learn anything about k
- (Π_1) If P_1 is honest, P_j cannot learn anything about m (except for its length)
- (Π_2) P_j cannot learn anything about m (except for its length)

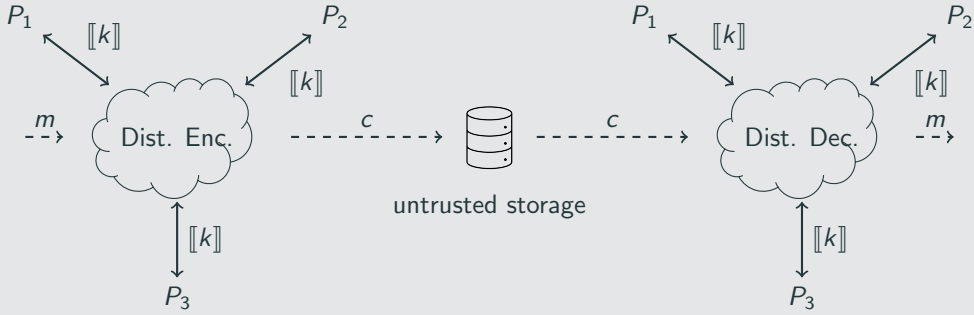
Authenticity

A corrupted party cannot forge a message

- A corrupted P_1 cannot make honest parties output m' given c'

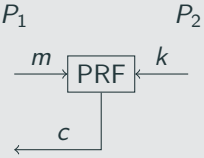
Applications

Protect Keys in Use: Encrypted Storage



$\rightarrow$ secret key k is shared among P_1, P_2, P_3

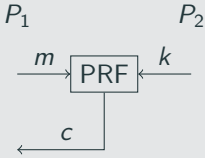
Oblivious PRFs



$\rightarrow$ private set intersection, pattern matching

$\rightarrow$ keyword search, secure database joins

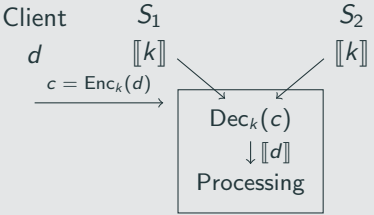
Oblivious PRFs



→ private set intersection, pattern matching

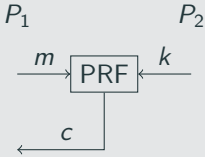
→ keyword search, secure database joins

Distributed Encryption/Decryption



→ transciphering, "encrypt to the future"

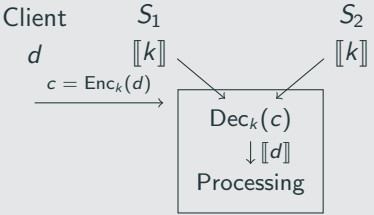
Oblivious PRFs



→ private set intersection, pattern matching

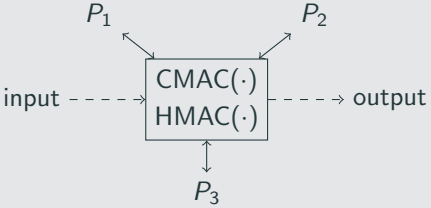
→ keyword search, secure database joins

Distributed Encryption/Decryption



→ transciphering, "encrypt to the future"

virtual hardware security module



→ protect cryptographic keys during use

→ decentralize Kerberos Key Distribution Center

→ distributed authentication/identification systems

NIST threshold cryptography project:
`https://csrc.nist.gov/projects/threshold-cryptography`

Distributed Symmetric-key Encryption (DiSE)

Distributed Symmetric-key Encryption (DiSE) [AMMR18]

Paper by Visa Research in CCS'18 with a few follow-up works

Main Idea:

1. Initiating party P_1 derives randomness α from m
2. All parties compute $w \leftarrow \mathsf{DPRF}(k, \alpha)$, P_1 obtains w
3. P_1 encrypts m with w
4. P_1 outputs $c = (\text{“Enc”}(w, m), \alpha)$

Distributed Pseudo-random Function (DPRF)

Syntax:

- Setup $\rightarrow [\![k]\!]_1, \ldots, [\![k]\!]_n, \mathsf{p}$
- $\mathsf{Eval}([\![k]\!]_i, x, \mathsf{p}) \rightarrow z^i$
- $\mathsf{Combine}(z^1, \ldots, z^s, \mathsf{p}) \rightarrow z$

Distributed Pseudo-random Function (DPRF)

Syntax:

- Setup $\rightarrow [\![k]\!]_1, \ldots, [\![k]\!]_n, \mathsf{p}$
- $\mathsf{Eval}([\![k]\!]_i, x, \mathsf{p}) \rightarrow z^i$
- $\mathsf{Combine}(z^1, \ldots, z^s, \mathsf{p}) \rightarrow z$

Properties

Pseudo-random

Output z is indistinguishable from output of a random function

Correctness

By sending $z^{j\prime}$, corrupted parties cannot bias z $\rightarrow$ honest parties either output z or abort

Commitment Scheme

Syntax:

- $\mathsf{Setup}_{\mathsf{Com}} \rightarrow \mathsf{p}_{\mathsf{Com}}$
- $\mathsf{Com}(x, r, \mathsf{p}_{\mathsf{Com}}) \rightarrow \alpha$

Commitment Scheme

Syntax:

- $\mathsf{Setup}_{\mathsf{Com}} \to \mathsf{p}_{\mathsf{Com}}$
- $\mathsf{Com}(x, r, \mathsf{p}_{\mathsf{Com}}) \to \alpha$

Properties

Hiding

Output α hides x (for random r)

Binding

The adversary cannot produce $(x_0, r_0) \neq (x_1, r_1)$ where $\mathsf{Com}(x_0, r_0, \mathsf{p}_{\mathsf{Com}}) = \mathsf{Com}(x_1, r_1, \mathsf{p}_{\mathsf{Com}})$

The Encryption Protocol

Setup: Run DPRF and commitment scheme setup;
P_i holds $\mathsf{p}, \mathsf{p}_{\mathsf{Com}}, [\![k]\!]_i$

The Encryption Protocol

Setup: Run DPRF and commitment scheme setup;
P_i holds $\mathsf{p}, \mathsf{p}_{\mathsf{Com}}, [\![k]\!]_i$

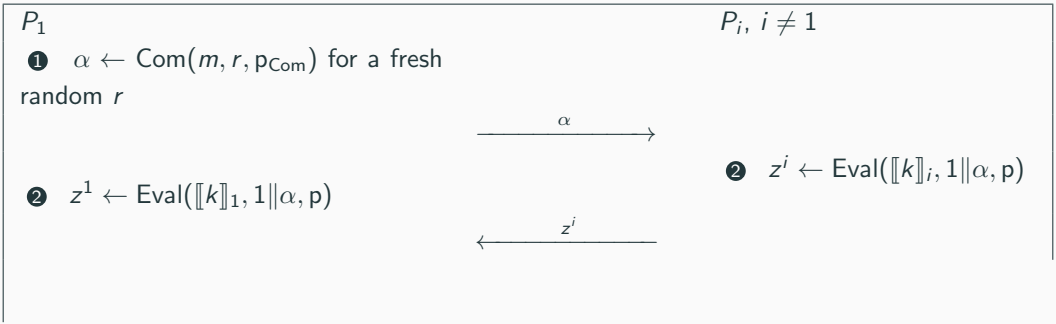
Message m known to P_1

P_1		P_i, $i \neq 1$
❶ $\alpha \leftarrow \mathsf{Com}(m, r, \mathsf{p}_{\mathsf{Com}})$ for a fresh random r	$\xrightarrow{\alpha}$	

The Encryption Protocol

Setup: Run DPRF and commitment scheme setup;
P_i holds $\mathsf{p}, \mathsf{p}_{\mathsf{Com}}, [\![k]\!]_i$

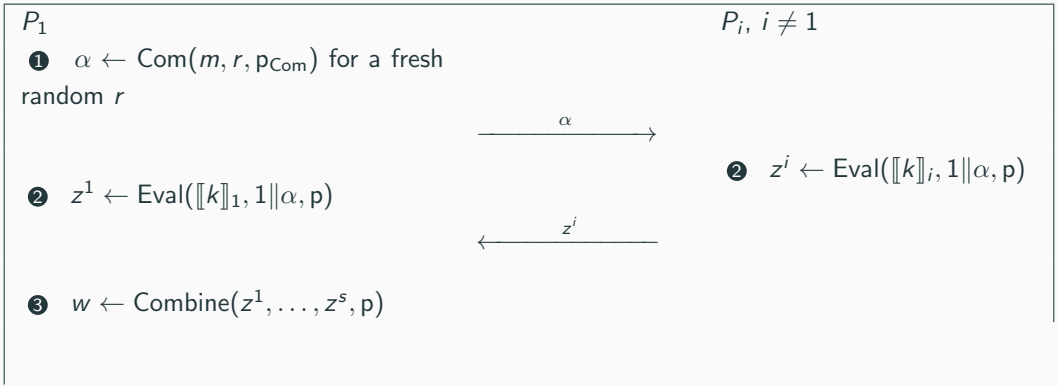
Message m known to P_1



The Encryption Protocol

Setup: Run DPRF and commitment scheme setup;
P_i holds $\mathsf{p}, \mathsf{p}_{\mathsf{Com}}, [\![k]\!]_i$

Message m known to P_1



The Encryption Protocol

Setup: Run DPRF and commitment scheme setup;
P_i holds $\mathsf{p}, \mathsf{p}_{\mathsf{Com}}, [\![k]\!]_i$

Message m known to P_1

P_1		$P_i,\ i \neq 1$
❶ $\alpha \leftarrow \mathsf{Com}(m, r, \mathsf{p}_{\mathsf{Com}})$ for a fresh random r	$\xrightarrow{\alpha}$	
		❷ $z^i \leftarrow \mathsf{Eval}([\![k]\!]_i, 1\|\|\alpha, \mathsf{p})$
❷ $z^1 \leftarrow \mathsf{Eval}([\![k]\!]_1, 1\|\|\alpha, \mathsf{p})$	$\xleftarrow{z^i}$	
❸ $w \leftarrow \mathsf{Combine}(z^1, \ldots, z^s, \mathsf{p})$		
❹ $e \leftarrow \mathsf{PRG}(w) \oplus (m\|\|r)$		
❺ output $c = (e, \alpha)$		

The Decryption Protocol

Setup: Run DPRF and commitment scheme setup;
P_i holds $\mathsf{p}, \mathsf{p}_{\mathsf{Com}}, [\![k]\!]_i$

Ciphertext c known to P_1

P_1		$P_i,\ i \neq 1$
❶ $(\alpha, e) = c$		
	$\xrightarrow{\alpha}$	
❷ $z^1 \leftarrow \mathsf{Eval}([\![k]\!]_1, 1\|\alpha, \mathsf{p})$		❷ $z^i \leftarrow \mathsf{Eval}([\![k]\!]_i, 1\|\alpha, \mathsf{p})$
	$\xleftarrow{z^i}$	
❸ $w \leftarrow \mathsf{Combine}(z^1, \ldots, z^s, \mathsf{p})$		
❹ $m'\|r' \leftarrow \mathsf{PRG}(w) \oplus e$		
❺ if $\alpha \neq \mathsf{Com}(m', r', \mathsf{p}_{\mathsf{Com}})$ abort		
❺ else: output $m = m'$		

Instantiating DPRF

for a threshold t-out-of-n sharing

Instantiating DPRF

for a threshold t-out-of-n sharing

PRF-based (Naor, Pinkas, Reingold)

Let F be a PRF.

- choose $k_1, \ldots, k_d$ at random
- key share $[\![k]\!]_i$ is replicated secret sharing using $k_1, \ldots, k_d$ as additive shares of the secret
- (after reconstruction) computes $F(k_1, x) \oplus \cdots \oplus F(k_d, x)$

Instantiating DPRF

for a threshold t-out-of-n sharing

PRF-based (Naor, Pinkas, Reingold)

Let F be a PRF.

- choose $k_1, \ldots, k_d$ at random
- key share $[\![k]\!]_i$ is replicated secret sharing using $k_1, \ldots, k_d$ as additive shares of the secret
- (after reconstruction) computes $F(k_1, x) \oplus \cdots \oplus F(k_d, x)$

DDH-based

Let G be a multiplicative cyclic group of prime order p where DDH assumption holds.
$\mathcal{H}(\cdot)$ hashes to G

- $[\![k]\!]_i$ is a Shamir sharing of a random $s \leftarrow\!\!\$\ \mathbb{Z}_p$
- $\mathrm{Eval}([\![k]\!]_i, x, \mathsf{p}) = \mathcal{H}(x)^{[\![k]\!]_i}$
- (after reconstruction) computes $\mathcal{H}(x)^s$.

$\rightarrow$ for $n \leq 24$ PRF-based DPRF is still faster despite exponential overhead

Performance

		Throughput (enc/s)	
n	t	PRF-based	DDH-based
4	2	1 113 090	555
6	2	1 095 770	556
18	6	45 434	297
24	16	445	100

numbers from Fig.9 [AMMR18]

MPC-based Distributed Symmetric-key Encryption

Another Approach: Using MPC

- focus on MPC evaluation of a block cipher
- use a block cipher in a mode of operation to encrypt (see tomorrow's talks)

Another Approach: Using MPC

- focus on MPC evaluation of a block cipher
- use a block cipher in a mode of operation to encrypt (see tomorrow's talks)

Block Cipher

A block cipher $F(k, m) \rightarrow c$ maps a b-bit input m under a λ-bit key to a b-bit output.

Another Approach: Using MPC

- focus on MPC evaluation of a block cipher
- use a block cipher in a mode of operation to encrypt (see tomorrow's talks)

Block Cipher

A block cipher $F(k, m) \rightarrow c$ maps a b-bit input m under a λ-bit key to a b-bit output.

Security?

A secure block cipher behaves like a (independent) pseudo-random permutation for each key

Another Approach: Using MPC

- focus on MPC evaluation of a block cipher
- use a block cipher in a mode of operation to encrypt (see tomorrow's talks)

Block Cipher

A block cipher $F(k, m) \rightarrow c$ maps a b-bit input m under a λ-bit key to a b-bit output.

Security?

A secure block cipher behaves like a (independent) pseudo-random permutation for each key

- Goal: construct MPC protocol to compute $F([\![k]\!], [\![m]\!]) \rightarrow c$ and $F^{-1}([\![k]\!], c) \rightarrow [\![m]\!]$.

Additive Secret Sharing

secret $x \in \mathrm{GF}(2^k)$:

- choose x_1, x_2, x_3 at random s.t. $x_1 + x_2 + x_3 = x$
- set $\langle x \rangle_i = x_i$ (for $i \in \{1, 2, 3\}$)

Additive Secret Sharing

secret $x \in \mathrm{GF}(2^k)$:

- choose x_1, x_2, x_3 at random s.t. $x_1 + x_2 + x_3 = x$
- set $\langle x \rangle_i = x_i$ (for $i \in \{1, 2, 3\}$)

Addition/Linear Operation

Addition of secret shared values $\langle x \rangle, \langle y \rangle$

- $\langle x + y \rangle_i = \langle x \rangle_i + \langle y \rangle_i$

Additive Secret Sharing

secret $x \in \mathrm{GF}(2^k)$:

- choose x_1, x_2, x_3 at random s.t. $x_1 + x_2 + x_3 = x$
- set $\langle x \rangle_i = x_i$ (for $i \in \{1, 2, 3\}$)

Addition/Linear Operation

Addition of secret shared values $\langle x \rangle, \langle y \rangle$

- $\langle x + y \rangle_i = \langle x \rangle_i + \langle y \rangle_i$ $\qquad \sum \langle x + y \rangle_i = (\sum \langle x \rangle_i) + (\sum \langle y \rangle_i) = x + y$

Additive Secret Sharing

secret $x \in \mathrm{GF}(2^k)$:

- choose x_1, x_2, x_3 at random s.t. $x_1 + x_2 + x_3 = x$
- set $\langle x \rangle_i = x_i$ (for $i \in \{1, 2, 3\}$)

Addition/Linear Operation

Addition of secret shared values $\langle x \rangle, \langle y \rangle$

- $\langle x + y \rangle_i = \langle x \rangle_i + \langle y \rangle_i$

$$\sum \langle x + y \rangle_i = \left(\sum \langle x \rangle_i\right) + \left(\sum \langle y \rangle_i\right) = x + y$$

Multiplication $\langle x \rangle$ by constant $a \in \mathrm{GF}(2^k)$

- $\langle a \cdot x \rangle_i = a \cdot \langle x \rangle_i$

Additive Secret Sharing

secret $x \in \mathrm{GF}(2^k)$:

- choose x_1, x_2, x_3 at random s.t. $x_1 + x_2 + x_3 = x$
- set $\langle x \rangle_i = x_i$ (for $i \in \{1, 2, 3\}$)

Addition/Linear Operation

Addition of secret shared values $\langle x \rangle, \langle y \rangle$

- $\langle x + y \rangle_i = \langle x \rangle_i + \langle y \rangle_i$ $\qquad \sum \langle x + y \rangle_i = (\sum \langle x \rangle_i) + (\sum \langle y \rangle_i) = x + y$

Multiplication $\langle x \rangle$ by constant $a \in \mathrm{GF}(2^k)$

- $\langle a \cdot x \rangle_i = a \cdot \langle x \rangle_i$ $\qquad \sum \langle a \cdot x \rangle_i = a \cdot (\sum \langle x \rangle_i) = a \cdot x$

Additive Secret Sharing

secret $x \in \mathrm{GF}(2^k)$:

- choose x_1, x_2, x_3 at random s.t. $x_1 + x_2 + x_3 = x$
- set $\langle x \rangle_i = x_i$ (for $i \in \{1, 2, 3\}$)

Addition/Linear Operation

Addition of secret shared values $\langle x \rangle, \langle y \rangle$

- $\langle x + y \rangle_i = \langle x \rangle_i + \langle y \rangle_i$

$$\sum \langle x + y \rangle_i = (\sum \langle x \rangle_i) + (\sum \langle y \rangle_i) = x + y$$

Multiplication $\langle x \rangle$ by constant $a \in \mathrm{GF}(2^k)$

- $\langle a \cdot x \rangle_i = a \cdot \langle x \rangle_i$

$$\sum \langle a \cdot x \rangle_i = a \cdot (\sum \langle x \rangle_i) = a \cdot x$$

Addition $\langle x \rangle$ with constant $b \in \mathrm{GF}(2^k)$

- $\langle x + b \rangle_1 = \langle x \rangle_1 + b$
- $\langle x + b \rangle_i = \langle x \rangle_i$ for $i \in \{2, 3\}$

Additive Secret Sharing

secret $x \in \mathrm{GF}(2^k)$:

- choose x_1, x_2, x_3 at random s.t. $x_1 + x_2 + x_3 = x$
- set $\langle x \rangle_i = x_i$ (for $i \in \{1, 2, 3\}$)

Addition/Linear Operation

Addition of secret shared values $\langle x \rangle, \langle y \rangle$

- $\langle x + y \rangle_i = \langle x \rangle_i + \langle y \rangle_i$

$$\sum \langle x + y \rangle_i = (\sum \langle x \rangle_i) + (\sum \langle y \rangle_i) = x + y$$

Multiplication $\langle x \rangle$ by constant $a \in \mathrm{GF}(2^k)$

- $\langle a \cdot x \rangle_i = a \cdot \langle x \rangle_i$

$$\sum \langle a \cdot x \rangle_i = a \cdot (\sum \langle x \rangle_i) = a \cdot x$$

Addition $\langle x \rangle$ with constant $b \in \mathrm{GF}(2^k)$

- $\langle x + b \rangle_1 = \langle x \rangle_1 + b$
- $\langle x + b \rangle_i = \langle x \rangle_i$ for $i \in \{2, 3\}$

$$\sum \langle x + b \rangle_i = (\sum \langle x \rangle) + b = x + b$$

Replicated Secret Sharing

Here: 3-party RSS (any two parties can reconstruct) $\rightarrow$ can be generalized to t-out-of-n

secret $x \in \mathrm{GF}(2^k)$:

Replicated Secret Sharing

Here: 3-party RSS (any two parties can reconstruct) $\rightarrow$ can be generalized to t-out-of-n

secret $x \in \mathrm{GF}(2^k)$:

- replicated sharing $[\![x]\!]_1 = (x_1, x_2)$, $[\![x]\!]_2 = (x_2, x_3)$ and $[\![x]\!]_3 = (x_3, x_1)$.

Replicated Secret Sharing

Here: 3-party RSS (any two parties can reconstruct) $\rightarrow$ can be generalized to t-out-of-n

secret $x \in \mathrm{GF}(2^k)$:

- replicated sharing $[\![x]\!]_1 = (x_1, x_2)$, $[\![x]\!]_2 = (x_2, x_3)$ and $[\![x]\!]_3 = (x_3, x_1)$.

Linear operation L: $[\![L(x)]\!]_i = (L(\langle x \rangle_i), L(\langle x \rangle_{i+1})) \rightarrow$ no communication

Squaring/other GF(2)-linear operations: also linear

Multiplication $[\![x]\!] \cdot [\![y]\!]$

Given $\langle 0 \rangle$ from preprocessing

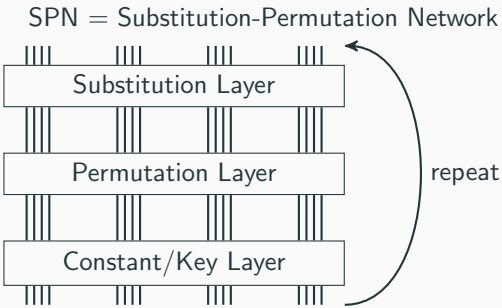
1. Local multiplication $\langle xy \rangle_i = x_i y_i + (x_i + x_{i+1})(y_i + y_{i+1})$ $\rightarrow$ obtain additive sharing of xy

Multiplication $[\![x]\!] \cdot [\![y]\!]$

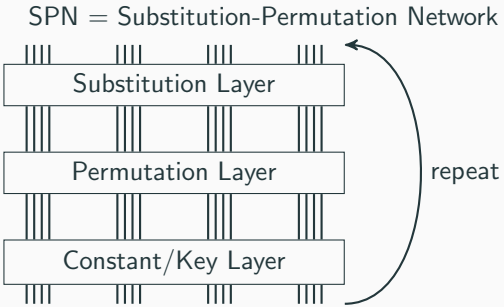
Given $\langle 0 \rangle$ from preprocessing

1. Local multiplication $\langle xy \rangle_i = x_i y_i + (x_i + x_{i+1})(y_i + y_{i+1})$ $\rightarrow$ obtain additive sharing of xy
2. Send $\langle xy \rangle_i + \langle 0 \rangle_i$ to P_{i-1}, set $[\![xy]\!]_i = (\langle xy \rangle_i + \langle 0 \rangle_i, \langle xy \rangle_{i+1} + \langle 0 \rangle_{i+1})$

Structure of a SPN Block Cipher

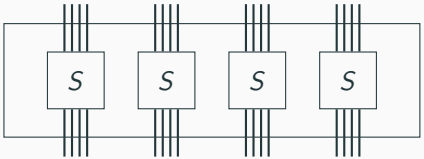


Structure of a SPN Block Cipher



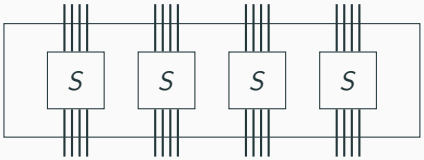
- "algorithm" structure
- substitution: non-linear
- permutation, constant/key layer: linear

The Substitution Layer



Substitution Layer

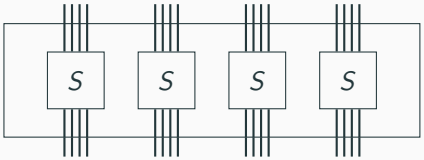
The Substitution Layer



Substitution Layer

S is a very non-linear s-bit to s-bit function (here: 4-bit to 4-bit)

The Substitution Layer

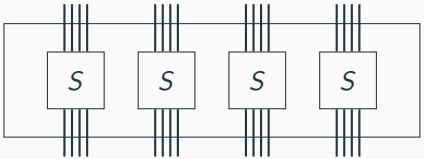


Substitution Layer

S is a very non-linear s-bit to s-bit function (here: 4-bit to 4-bit)

How to represent it with secret-independent control flow?

The Substitution Layer



Substitution Layer

S is a very non-linear s-bit to s-bit function (here: 4-bit to 4-bit)

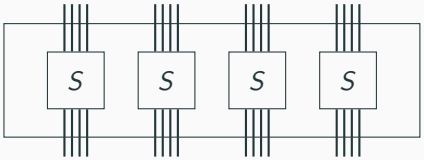
How to represent it with secret-independent control flow?

A couple of approaches:

represent S

1. as Boolean circuit
2. as (uni-variate) polynomial
3. as oblivious table lookup

The Substitution Layer



Substitution Layer

S is a very non-linear s-bit to s-bit function (here: 4-bit to 4-bit)

How to represent it with secret-independent control flow?

A couple of approaches:

represent S

1. as Boolean circuit
2. as (uni-variate) polynomial
3. as oblivious table lookup

similar approaches in literature for constant-time / software side-channel protection for block ciphers!

Useful/MPC-friendly Properties

What properties should the approach have?

- Low multiplicative complexity:
  - low total number of multiplications/AND gates
  - $R \times n_S \times mult_S$

Useful/MPC-friendly Properties

What properties should the approach have?

- Low multiplicative complexity:
  - low total number of multiplications/AND gates
  - $R \times n_S \times mult_S$
- Low multiplicative depth:
  - round-based structure: $R \times rounds_S$ in total

AES as Example

Prelude: Galois Fields

We write $\mathrm{GF}(2^k) = \mathbb{F}_2[X]/F(X)$

- $F(X)$ irreducible polynomial of degree k: $F(X) = X^4 + X + 1$
- element $x \in \mathrm{GF}(2^k)$: polynomial of degree $< k$: $X^2 + X + 1$

Prelude: Galois Fields

We write $\mathrm{GF}(2^k) = \mathbb{F}_2[X]/F(X)$

- $F(X)$ irreducible polynomial of degree k: $F(X) = X^4 + X + 1$
- element $x \in \mathrm{GF}(2^k)$: polynomial of degree $< k$: $X^2 + X + 1$

Addition

coefficient-wise addition in $\mathbb{F}_2$

$$(X^3 + 1) + (X^2 + X + 1) = X^3 + X^2 + X$$

Multiplication

polynomial multiplication modulo $F(X)$
schoolbook division: $(X^3+1)\cdot(X^2+X+1) = (X^5+X^4+X^3+X^2+X+1) \mod F(X)$

Multiplication

polynomial multiplication modulo $F(X)$

schoolbook division: $(X^3+1)\cdot(X^2+X+1) = (X^5+X^4+X^3+X^2+X+1) \mod F(X)$

$$
\begin{array}{rrrrrrl}
X^5 & +X^4 & +X^3 & +X^2 & +X & +1 & /(X^4+X+1) = X+1 \\
-X^5 & & & -X^2 & -X & & \\
\hline
 & X^4 & +X^3 & & & +1 & \\
 & -X^4 & & & -X & -1 & \\
\hline
 & & X^3 & & +X & &
\end{array}
$$

Multiplication

polynomial multiplication modulo $F(X)$

schoolbook division: $(X^3+1)\cdot(X^2+X+1) = (X^5+X^4+X^3+X^2+X+1) \mod F(X)$

$$
\begin{array}{rrrrrrl}
X^5 & +X^4 & +X^3 & +X^2 & +X & +1 & /(X^4+X+1) = X+1 \\
-X^5 & & & -X^2 & -X & & \\
\hline
 & X^4 & +X^3 & & & +1 & \\
 & -X^4 & & & -X & -1 & \\
\hline
 & & X^3 & & +X & &
\end{array}
$$

$\rightarrow (X^3+1)\cdot(X^2+X+1) \mod F(X) = X^3+X$

Multiplication

polynomial multiplication modulo $F(X)$

schoolbook division: $(X^3+1)\cdot(X^2+X+1) = (X^5+X^4+X^3+X^2+X+1) \mod F(X)$

$$
\begin{array}{rrrrrrl}
X^5 & +X^4 & +X^3 & +X^2 & +X & +1 & /(X^4+X+1) = X+1 \\
-X^5 & & & -X^2 & -X & & \\
\hline
 & X^4 & +X^3 & & & +1 & \\
 & -X^4 & & & -X & -1 & \\
\hline
 & & X^3 & & +X & &
\end{array}
$$

$\rightarrow (X^3+1)\cdot(X^2+X+1) \mod F(X) = X^3+X$

We write binary encoding: $13 = 2^3+2^2+2^0 = (X^3+X^2+1)$

The AES

- AES = Advanced Encryption Standard
- Rijndael block cipher standardized by NIST 2001
- block size: 128-bit
- key sizes: **128-bit**, 192-bit, 256-bit
- Block cipher composed of 10 rounds (AES-128)
- defined over $\mathrm{GF}(2^8) = \mathbb{F}_2[X]/X^8 + X^4 + X^3 + X + 1$.

- 128-bit Input: $b_0 b_1 \dots b_{127}$
  - group by 8: $\underbrace{(b_0 b_1 \dots b_7)}_{s_0}\ \underbrace{(b_8 \dots)}_{s_1}\ \dots\ \underbrace{(\dots b_{127})}_{s_{15}}$

- 128-bit Input: $b_0 b_1 \dots b_{127}$
  - group by 8: $\underbrace{(b_0 b_1 \dots b_7)}_{s_0}\ \underbrace{(b_8 \dots)}_{s_1}\ \dots\ \underbrace{(\dots b_{127})}_{s_{15}}$
  - arrange column-first in 4×4 matrix

$$\text{state} = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$$

- 128-bit Input: $b_0 b_1 \dots b_{127}$
  - group by 8: $\underbrace{(b_0 b_1 \dots b_7)}_{s_0}\ \underbrace{(b_8 \dots)}_{s_1}\ \dots\ \underbrace{(\dots b_{127})}_{s_{15}}$
  - arrange column-first in 4×4 matrix

$$\text{state} = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$$

- Key Schedule takes 128-bit key: produces 11 round keys

$$\underbrace{\begin{pmatrix} rk_0^{(0)} & rk_4^{(0)} & rk_8^{(0)} & rk_{12}^{(0)} \\ rk_1^{(0)} & rk_5^{(0)} & rk_9^{(0)} & rk_{13}^{(0)} \\ rk_2^{(0)} & rk_6^{(0)} & rk_{10}^{(0)} & rk_{14}^{(0)} \\ rk_3^{(0)} & rk_7^{(0)} & rk_{11}^{(0)} & rk_{15}^{(0)} \end{pmatrix}}_{RK_0} \quad \dots \quad \underbrace{\begin{pmatrix} rk_0^{(10)} & rk_4^{(10)} & rk_8^{(10)} & rk_{12}^{(10)} \\ rk_1^{(10)} & rk_5^{(10)} & rk_9^{(10)} & rk_{13}^{(10)} \\ rk_2^{(10)} & rk_6^{(10)} & rk_{10}^{(10)} & rk_{14}^{(10)} \\ rk_3^{(10)} & rk_7^{(10)} & rk_{11}^{(10)} & rk_{15}^{(10)} \end{pmatrix}}_{RK_{10}}$$

AES Round Function

1. SubBytes: S-box$(s_i) = f(s_i^{-1})$

AES Round Function

1. SubBytes: $\text{S-box}(s_i) = f(s_i^{-1})$
2. ShiftRows: $\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix} \rightarrow \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_5 & s_9 & s_{13} & s_1 \\ s_{10} & s_{14} & s_2 & s_6 \\ s_{15} & s_3 & s_7 & s_{11} \end{pmatrix}$

AES Round Function

1. SubBytes: $\text{S-box}(s_i) = f(s_i^{-1})$
2. ShiftRows: $\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix} \rightarrow \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_5 & s_9 & s_{13} & s_1 \\ s_{10} & s_{14} & s_2 & s_6 \\ s_{15} & s_3 & s_7 & s_{11} \end{pmatrix}$
3. MixColumns: $\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$

AES Round Function

1. SubBytes: $\text{S-box}(s_i) = f(s_i^{-1})$
2. ShiftRows: $\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix} \rightarrow \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_5 & s_9 & s_{13} & s_1 \\ s_{10} & s_{14} & s_2 & s_6 \\ s_{15} & s_3 & s_7 & s_{11} \end{pmatrix}$
3. MixColumns: $\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$
4. AddRoundKey: $\text{state} + RK_r$

AES S-box as Boolean Circuit

due to Boyar and Peralta [BP10]: 32 AND gates, 83 XOR gates

AES S-box as Boolean Circuit

due to Boyar and Peralta [BP10]: 32 AND gates, 83 XOR gates

$$
\begin{array}{lll}
y_{14} = x_3 + x_5 & y_{13} = x_0 + x_6 & y_9 = x_0 + x_3 \\
y_8 = x_0 + x_5 & t_0 = x_1 + x_2 & y_1 = t_0 + x_7 \\
y_4 = y_1 + x_3 & y_{12} = y_{13} + y_{14} & y_2 = y_1 + x_0 \\
y_5 = y_1 + x_6 & y_3 = y_5 + y_8 & t_1 = x_4 + y_{12} \\
y_{15} = t_1 + x_5 & y_{20} = t_1 + x_1 & y_6 = y_{15} + x_7 \\
y_{10} = y_{15} + t_0 & y_{11} = y_{20} + y_9 & y_7 = x_7 + y_{11} \\
y_{17} = y_{10} + y_{11} & y_{19} = y_{10} + y_8 & y_{16} = t_0 + y_{11} \\
y_{21} = y_{13} + y_{16} & y_{18} = x_0 + y_{16} &
\end{array}
$$

$$
\begin{array}{lll}
t_{46} = z_{15} + z_{16} & t_{47} = z_{10} + z_{11} & t_{48} = z_5 + z_{13} \\
t_{49} = z_9 + z_{10} & t_{50} = z_2 + z_{12} & t_{51} = z_2 + z_5 \\
t_{52} = z_7 + z_8 & t_{53} = z_0 + z_3 & t_{54} = z_6 + z_7 \\
t_{55} = z_{16} + z_{17} & t_{56} = z_{12} + t_{48} & t_{57} = t_{50} + t_{53} \\
t_{58} = z_4 + t_{46} & t_{59} = z_3 + t_{54} & t_{60} = t_{46} + t_{57} \\
t_{61} = z_{14} + t_{57} & t_{62} = t_{52} + t_{58} & t_{63} = t_{49} + t_{58} \\
t_{64} = z_4 + t_{59} & t_{65} = t_{61} + t_{62} & t_{66} = z_1 + t_{63} \\
s_0 = t_{59} + t_{63} & s_6 = t_{56} \text{ XNOR } t_{62} & s_7 = t_{48} \text{ XNOR } t_{60} \\
t_{67} = t_{64} + t_{65} & s_3 = t_{53} + t_{66} & s_4 = t_{51} + t_{66} \\
s_5 = t_{47} + t_{65} & s_1 = t_{64} \text{ XNOR } s_3 & s_2 = t_{55} \text{ XNOR } t_{67}
\end{array}
$$

$$
\begin{array}{lll}
t_2 = y_{12} \times y_{15} & t_3 = y_3 \times y_6 & t_4 = t_3 + t_2 \\
t_5 = y_4 \times x_7 & t_6 = t_5 + t_2 & t_7 = y_{13} \times y_{16} \\
t_8 = y_5 \times y_1 & t_9 = t_8 + t_7 & t_{10} = y_2 \times y_7 \\
t_{11} = t_{10} + t_7 & t_{12} = y_9 \times y_{11} & t_{13} = y_{14} \times y_{17} \\
t_{14} = t_{13} + t_{12} & t_{15} = y_8 \times y_{10} & t_{16} = t_{15} + t_{12} \\
t_{17} = t_4 + t_{14} & t_{18} = t_6 + t_{16} & t_{19} = t_9 + t_{14} \\
t_{20} = t_{11} + t_{16} & t_{21} = t_{17} + y_{20} & t_{22} = t_{18} + y_{19} \\
t_{23} = t_{19} + y_{21} & t_{24} = t_{20} + y_{18} &
\end{array}
$$

$$
\begin{array}{lll}
t_{25} = t_{21} + t_{22} & t_{26} = t_{21} \times t_{23} & t_{27} = t_{24} + t_{26} \\
t_{28} = t_{25} \times t_{27} & t_{29} = t_{28} + t_{22} & t_{30} = t_{23} + t_{24} \\
t_{31} = t_{22} + t_{26} & t_{32} = t_{31} \times t_{30} & t_{33} = t_{32} + t_{24} \\
t_{34} = t_{23} + t_{33} & t_{35} = t_{27} + t_{33} & t_{36} = t_{24} \times t_{35} \\
t_{37} = t_{36} + t_{34} & t_{38} = t_{27} + t_{36} & t_{39} = t_{29} \times t_{38} \\
t_{40} = t_{25} + t_{39} &
\end{array}
$$

$$
\begin{array}{lll}
t_{41} = t_{40} + t_{37} & t_{42} = t_{29} + t_{33} & t_{43} = t_{29} + t_{40} \\
t_{44} = t_{33} + t_{37} & t_{45} = t_{42} + t_{41} & z_0 = t_{44} \times y_{15} \\
z_1 = t_{37} \times y_6 & z_2 = t_{33} \times x_7 & z_3 = t_{43} \times y_{16} \\
z_4 = t_{40} \times y_1 & z_5 = t_{29} \times y_7 & z_6 = t_{42} \times y_{11} \\
z_7 = t_{45} \times y_{17} & z_8 = t_{41} \times y_{10} & z_9 = t_{44} \times y_{12} \\
z_{10} = t_{37} \times y_3 & z_{11} = t_{33} \times y_4 & z_{12} = t_{43} \times y_{13} \\
z_{13} = t_{40} \times y_5 & z_{14} = t_{29} \times y_2 & z_{15} = t_{42} \times y_9 \\
z_{16} = t_{45} \times y_{14} & z_{17} = t_{41} \times y_8 &
\end{array}
$$

$\rightarrow$ in circuit form: `https://nigelsmart.github.io/MPC-Circuits/`

Cost: 32 GF(2) multiplications in 6 rounds

AES S-box as Polynomial

Goal: find $P(Z)$ s.t.

$$
\begin{aligned}
P(0) &= f(0) \\
P(1) &= f(1) \\
P(2) &= f(2^{-1}) \\
&\vdots \\
P(255) &= f(255^{-1})
\end{aligned}
$$

AES S-box as Polynomial

Goal: find $P(Z)$ s.t.

$$
\begin{aligned}
P(0) &= f(0) \\
P(1) &= f(1) \\
P(2) &= f(2^{-1}) \\
&\vdots \\
P(255) &= f(255^{-1})
\end{aligned}
$$

Approach 1: due to [DKLMS12]

$$
\begin{aligned}
P(z) = \mathrm{0x63} &+ \mathrm{0x8F}z^{127} + \mathrm{0xB5}z^{191} + z^{223} + \mathrm{0xF4}z^{239} + \mathrm{0x25}z^{247} \\
&+ \mathrm{0xF9}z^{251} + \mathrm{0x09}z^{253} + \mathrm{0x05}z^{254}
\end{aligned}
$$

AES S-box as Polynomial

Goal: find $P(Z)$ s.t.

$$\begin{aligned} P(0) &= f(0) \\ P(1) &= f(1) \\ P(2) &= f(2^{-1}) \\ &\vdots \\ P(255) &= f(255^{-1}) \end{aligned}$$

Approach 1: due to [DKLMS12]

$$\begin{aligned} P(z) = \texttt{0x63} + \texttt{0x8F}z^{127} + \texttt{0xB5}z^{191} + z^{223} + \texttt{0xF4}z^{239} + \texttt{0x25}z^{247} \\ + \texttt{0xF9}z^{251} + \texttt{0x09}z^{253} + \texttt{0x05}z^{254} \end{aligned}$$

Cost: shortest addition chain for the set $\{127, 191, 223, 239, 247, 251, 253, 254\}$

- 18 secure multiplications in $\mathrm{GF}(2^8)$ in 12 rounds

Approach 2: Using BitDecomposition

BitDecomposition: $[\![b_7X^7 + \cdots + b_1X + b_0]\!] \rightarrow [\![b_7]\!], \ldots, [\![b_1]\!], [\![b_0]\!]$
$\rightarrow$ computing $z^2 \in \mathrm{GF}(2^8)$ is bit-linear!

Approach 2: Using BitDecomposition

BitDecomposition: $[\![b_7 X^7 + \cdots + b_1 X + b_0]\!] \to [\![b_7]\!], \ldots, [\![b_1]\!], [\![b_0]\!]$
$\to$ computing $z^2 \in \mathrm{GF}(2^8)$ is bit-linear!

- Observation: $f(z^{-1}) = f(z^{254})$ (since 0 is mapped to 0)

Approach 2: Using BitDecomposition

BitDecomposition: $[\![b_7X^7 + \cdots + b_1X + b_0]\!] \to [\![b_7]\!], \ldots, [\![b_1]\!], [\![b_0]\!]$
$\to$ computing $z^2 \in \mathrm{GF}(2^8)$ is bit-linear!

- Observation: $f(z^{-1}) = f(z^{254})$ (since 0 is mapped to 0)
- write z^{254} as
  1. $z^3 = z \cdot z^2$ (1 mult.)
  2. $z^{15} = (z^3)^4 \cdot z^3$, $z^{14} = (z^3)^4 \cdot z^2$ (2 mult.)
  3. $z^{254} = (z^{15})^{16} \cdot z^{14}$ (1 mult.)

Approach 2: Using BitDecomposition

BitDecomposition: $[\![b_7 X^7 + \cdots + b_1 X + b_0]\!] \to [\![b_7]\!], \ldots, [\![b_1]\!], [\![b_0]\!]$
$\to$ computing $z^2 \in \mathrm{GF}(2^8)$ is bit-linear!

- Observation: $f(z^{-1}) = f(z^{254})$ (since 0 is mapped to 0)
- write z^{254} as
  1. $z^3 = z \cdot z^2$ (1 mult.)
  2. $z^{15} = (z^3)^4 \cdot z^3$, $z^{14} = (z^3)^4 \cdot z^2$ (2 mult.)
  3. $z^{254} = (z^{15})^{16} \cdot z^{14}$ (1 mult.)

Cost: 4 $\mathrm{GF}(2^8)$ multiplications in 3 rounds

cf. [DK10], [CHIKP18]

Approach 3: Using Tower Fields [MPSSTT25]

Isomorphism $\phi : \mathrm{GF}(2^8) \to \mathrm{GF}(2^4)^2$

1. $\phi(z) = (a_h, a_\ell)$

Approach 3: Using Tower Fields [MPSSTT25]

Isomorphism $\phi : \mathrm{GF}(2^8) \to \mathrm{GF}(2^4)^2$

1. $\phi(z) = (a_h, a_\ell)$
2. $v = \mathrm{e}a_h^2 + a_h \cdot a_\ell + a_\ell^2$ (1 mult.)
3. $v^{-1} = v^2 \cdot v^4 \cdot v^8$ (2 mult.)

Approach 3: Using Tower Fields [MPSSTT25]

Isomorphism $\phi : \mathrm{GF}(2^8) \to \mathrm{GF}(2^4)^2$

1. $\phi(z) = (a_h, a_\ell)$
2. $v = \mathrm{e}a_h^2 + a_h \cdot a_\ell + a_\ell^2$ (1 mult.)
3. $v^{-1} = v^2 \cdot v^4 \cdot v^8$ (2 mult.)
4. $z^{-1} = \phi^{-1}(a_h \cdot v^{-1}, (a_h + a_\ell) \cdot v^{-1})$ (2 mult.)

Approach 3: Using Tower Fields [MPSSTT25]

Isomorphism $\phi : \mathrm{GF}(2^8) \to \mathrm{GF}(2^4)^2$

1. $\phi(z) = (a_h, a_\ell)$
2. $v = \mathrm{e}a_h^2 + a_h \cdot a_\ell + a_\ell^2$ (1 mult.)
3. $v^{-1} = v^2 \cdot v^4 \cdot v^8$ (2 mult.)
4. $z^{-1} = \phi^{-1}(a_h \cdot v^{-1}, (a_h + a_\ell) \cdot v^{-1})$ (2 mult.)

Cost: 5 $\mathrm{GF}(2^4)$ multiplications in 4 rounds

Oblivious Table Lookup Protocol

Goal: compute table lookup LUT $[\![T[x]]\!]$ from table T and $[\![x]\!]$

Oblivious Table Lookup Protocol

Goal: compute table lookup LUT $[\![T[x]]\!]$ from table T and $[\![x]\!]$

given *random one-hot vector* correlation from preprocessing $[\![r]\!], [\![\boldsymbol{e}^{(r)}]\!]$

$$[\![\boldsymbol{e}^{(r)}]\!] = (\underbrace{[\![0]\!], \ldots, [\![0]\!], [\![1]\!]}_{r}, [\![0]\!], \ldots)$$

Oblivious Table Lookup Protocol

Goal: compute table lookup LUT $[\![T[x]]\!]$ from table T and $[\![x]\!]$

given *random one-hot vector* correlation from preprocessing $[\![r]\!], [\![\boldsymbol{e}^{(r)}]\!]$

$$[\![\boldsymbol{e}^{(r)}]\!] = (\underbrace{[\![0]\!], \ldots, [\![0]\!], [\![1]\!]}_{r}, [\![0]\!], \ldots)$$

1. Reveal value $c = r + x$
2. Local computation $[\![T[x]]\!] = \sum_i T[c - i] \cdot [\![e_i^{(r)}]\!]$

AES S-box as Oblivious Table Lookup

AES S-box table: 256 elements

$\rightarrow$ costly preprocessing of length-256 random one-hot vector ($\approx$ 247 bit-multiplications)

AES S-box as Oblivious Table Lookup

AES S-box table: 256 elements

$\rightarrow$ costly preprocessing of length-256 random one-hot vector ($\approx$ 247 bit-multiplications)

MAESTRO (3-party, honest majority) [MPSSTT25]

using replicated secret sharing reduce cost with a trick
Better preprocessing:

AES S-box as Oblivious Table Lookup

AES S-box table: 256 elements

$\rightarrow$ costly preprocessing of length-256 random one-hot vector ($\approx$ 247 bit-multiplications)

MAESTRO (3-party, honest majority) [MPSSTT25]

using replicated secret sharing reduce cost with a trick

Better preprocessing:

1. obtain 2 length-16 random one-hot vectors: $[\![u]\!], [\![\boldsymbol{e}^{(u)}]\!]$ and $[\![v]\!], [\![\boldsymbol{e}^{(v)}]\!]$

AES S-box as Oblivious Table Lookup

AES S-box table: 256 elements

$\rightarrow$ costly preprocessing of length-256 random one-hot vector ($\approx$ 247 bit-multiplications)

MAESTRO (3-party, honest majority) [MPSSTT25]

using replicated secret sharing reduce cost with a trick

Better preprocessing:

1. obtain 2 length-16 random one-hot vectors: $[\![u]\!], [\![\boldsymbol{e}^{(u)}]\!]$ and $[\![v]\!], [\![\boldsymbol{e}^{(v)}]\!]$
2. *local* tensor product $\langle \boldsymbol{e}^{(r)} \rangle \leftarrow [\![\boldsymbol{e}^{(u)}]\!] \otimes [\![\boldsymbol{e}^{(v)}]\!]$
   where $r = u \| v$

AES S-box as Oblivious Table Lookup

AES S-box table: 256 elements

$\rightarrow$ costly preprocessing of length-256 random one-hot vector ($\approx$ 247 bit-multiplications)

MAESTRO (3-party, honest majority) [MPSSTT25]

using replicated secret sharing reduce cost with a trick

Better preprocessing:

1. obtain 2 length-16 random one-hot vectors: $[\![u]\!], [\![\boldsymbol{e}^{(u)}]\!]$ and $[\![v]\!], [\![\boldsymbol{e}^{(v)}]\!]$
2. *local* tensor product $\langle \boldsymbol{e}^{(r)} \rangle \leftarrow [\![\boldsymbol{e}^{(u)}]\!] \otimes [\![\boldsymbol{e}^{(v)}]\!]$
   where $r = u \| v$
3. output $([\![r]\!], \langle \boldsymbol{e}^{(r)} \rangle)$

AES S-box as Oblivious Table Lookup

AES S-box table: 256 elements

$\rightarrow$ costly preprocessing of length-256 random one-hot vector ($\approx$ 247 bit-multiplications)

MAESTRO (3-party, honest majority) [MPSSTT25]

using replicated secret sharing reduce cost with a trick

Better preprocessing:

1. obtain 2 length-16 random one-hot vectors: $[\![u]\!], [\![\boldsymbol{e}^{(u)}]\!]$ and $[\![v]\!], [\![\boldsymbol{e}^{(v)}]\!]$
2. *local* tensor product $\langle \boldsymbol{e}^{(r)} \rangle \leftarrow [\![\boldsymbol{e}^{(u)}]\!] \otimes [\![\boldsymbol{e}^{(v)}]\!]$
   where $r = u \| v$
3. output $([\![r]\!], \langle \boldsymbol{e}^{(r)} \rangle)$

$\rightarrow$ cost: 22 GF(2) multiplications in 2 rounds (preprocessing)

$\rightarrow$ cost: 16 bits in 1 round (online)

Performance

→ measure throughput in AES blocks/s

	Throughput (blocks/s)					
Protocol	Online	Total	Online	Total	Online	Total
Polynomial Approach 2	568 504	568 504	124 290	124 290	13 392	13 392
Polynomial Approach 3	729 822	**729 822**	179 369	**179 369**	**18 497**	**18 497**
Oblv. Table Lookup	641 302	108 114	148 917	39 156	16 574	5 216
Network	10 Gbit/s $\leq$ 1ms RTT		1 Gbit/s $\leq$ 1ms RTT		100 MBit/s 30ms RTT	

semi-honest security, 16-core 128GB RAM

$\rightarrow$ measure throughput in AES blocks/s

	Throughput (blocks/s)					
Protocol	Online	Total	Online	Total	Online	Total
Polynomial Approach 2*	**152 127**	10 100	**33 646**	4 490	4 707	1 039
Polynomial Approach 3	48 924	**48 924**	18 829	**18 829**	**5 396**	**5 396**
Oblv. Table Lookup	32 508	25 409	13 547	10 560	3 370	2 345
Network	10 Gbit/s $\leq$ 1ms RTT		1 Gbit/s $\leq$ 1ms RTT		100 MBit/s 30ms RTT	

* with [FLNW17]

active security, 16-core 128GB RAM

Thank you!

References

[AMMR18] Agrawal, Mohassel, Mukherjee and Rindal, DiSE: Distributed Symmetric-key Encryption, *CCS*, 2018.

[BP10] Boyar and Peralta, A new combinational logic minimization technique with applications to cryptology, *International Symposium on Experimental Algorithms*, 2010.

[CHIKP18] Chida, Hamada, Ikarashi, Kikuchi and Pinkas, High-Throughput Secure AES Computation, *WAHC*, 2018.

[DK10] Damgård and Keller, Secure Multiparty AES, *FC*, 2010.

[DKLMS12] Damgård, Keller, Larraia, Miles and Smart, Implementing AES via an Actively/Covertly Secure Dishonest-Majority MPC Protocol, *SCN*, 2012.

[FLNW17] Furukawa, Lindell, Nof and Weinstein, High-throughput secure three-party computation for malicious adversaries and an honest majority, *Eurocrypt*, 2017.

[MPSSTT25] Morita, Pohle, Sadakane, Scholl, Tozawa and Tschudi, MAESTRO: Multi-Party AES Using Lookup Tables, *Usenix Security*, 2025.