On the history of FHEMPCZK-friendly symmetric crypto

Christian Rechberger, TU Graz

LeoCrypt23

A Zoo of FHEMPCZK-friendly concretely-efficient symmetric crypto: How many designs?

2013: -

2014: -

2015: 1

2016: 4

2017: -

2018: 3

2019: 5

2020: 5

2021: 8

2022: 10

2023: 4 until April

source: mostly IACR eprint, plus selection from IEEE Access, ToSC, arxiv

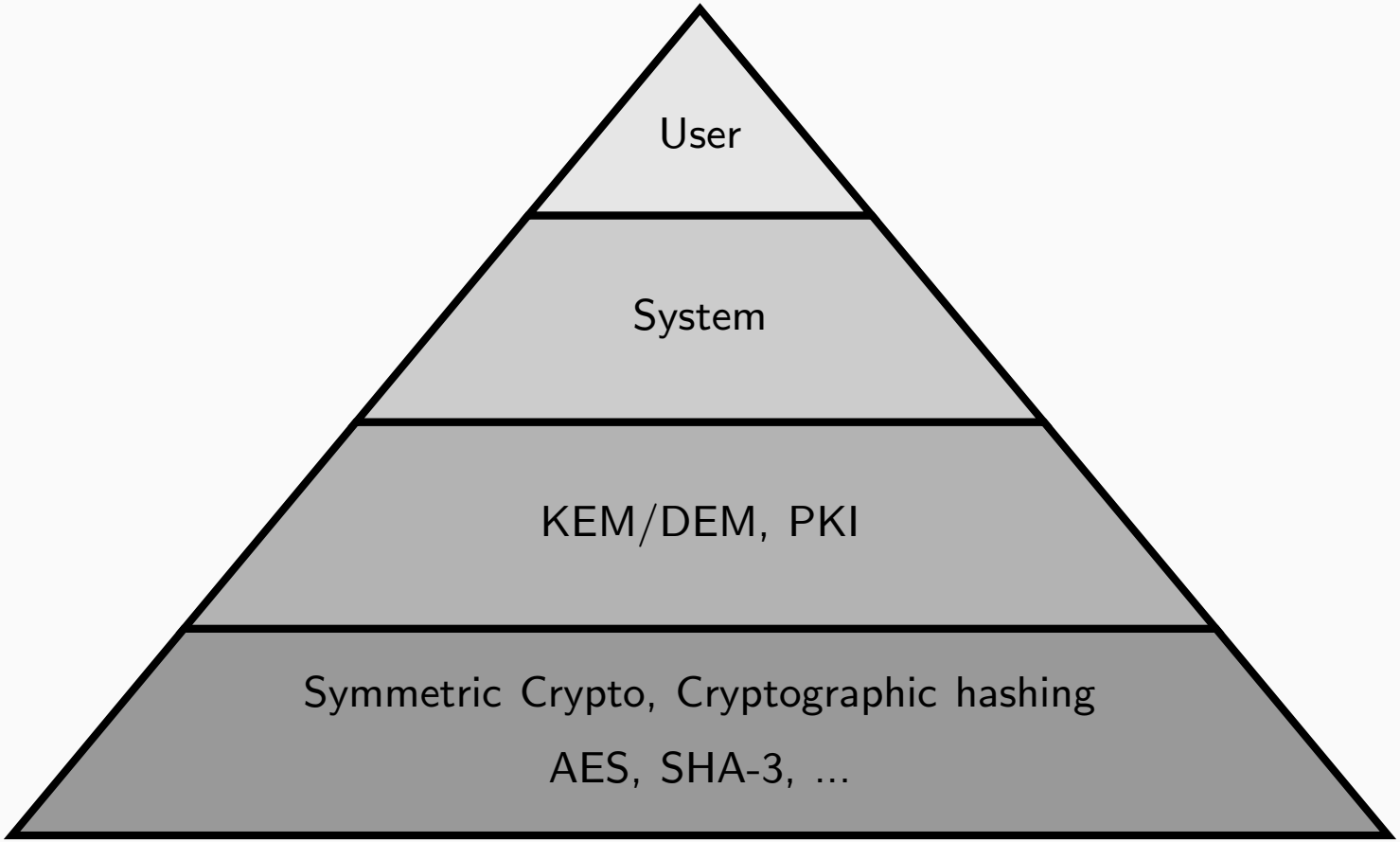
How did we get here?

Implementation environments for symmetric cryptography

Efficiently provide confidentiality, authenticity, integrity

- until 1980s: dedicated machines, hardware implementing DES, LFSR-based approaches
- since 1990s: software implementations become more relevant in addition to hardware, see e.g. AES
- since 2010s: another boost for software-environments due to virtualization

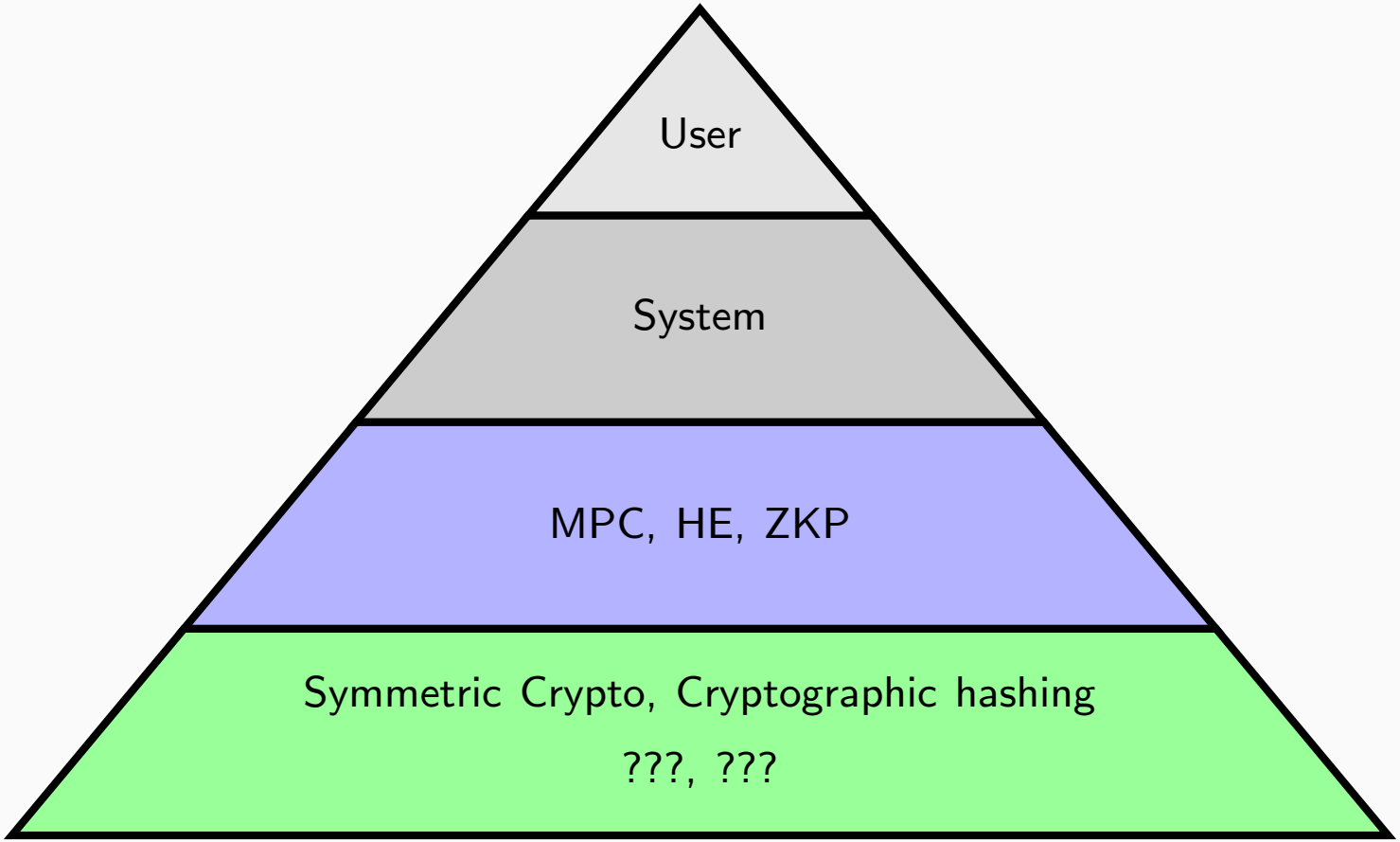
Role of symmetric-key crypto and hashing in systems



New cryptographic functionalities are new applications of symmetric cryptography

- FHE: Reducing ciphertext expansion, OPRFs, ...
- MPC: Distributed databases, private set intersection, data analytics, but also new public-key signature schemes
- ZKP: Use-cases of zero-knowledge proofs:
  - Set Membership Proofs ("I know a private key of one of the public keys of this Merkle tree")
  - Data Commitments ("Here is the Merkle tree of the execution trace of my program, I can open it at any point").

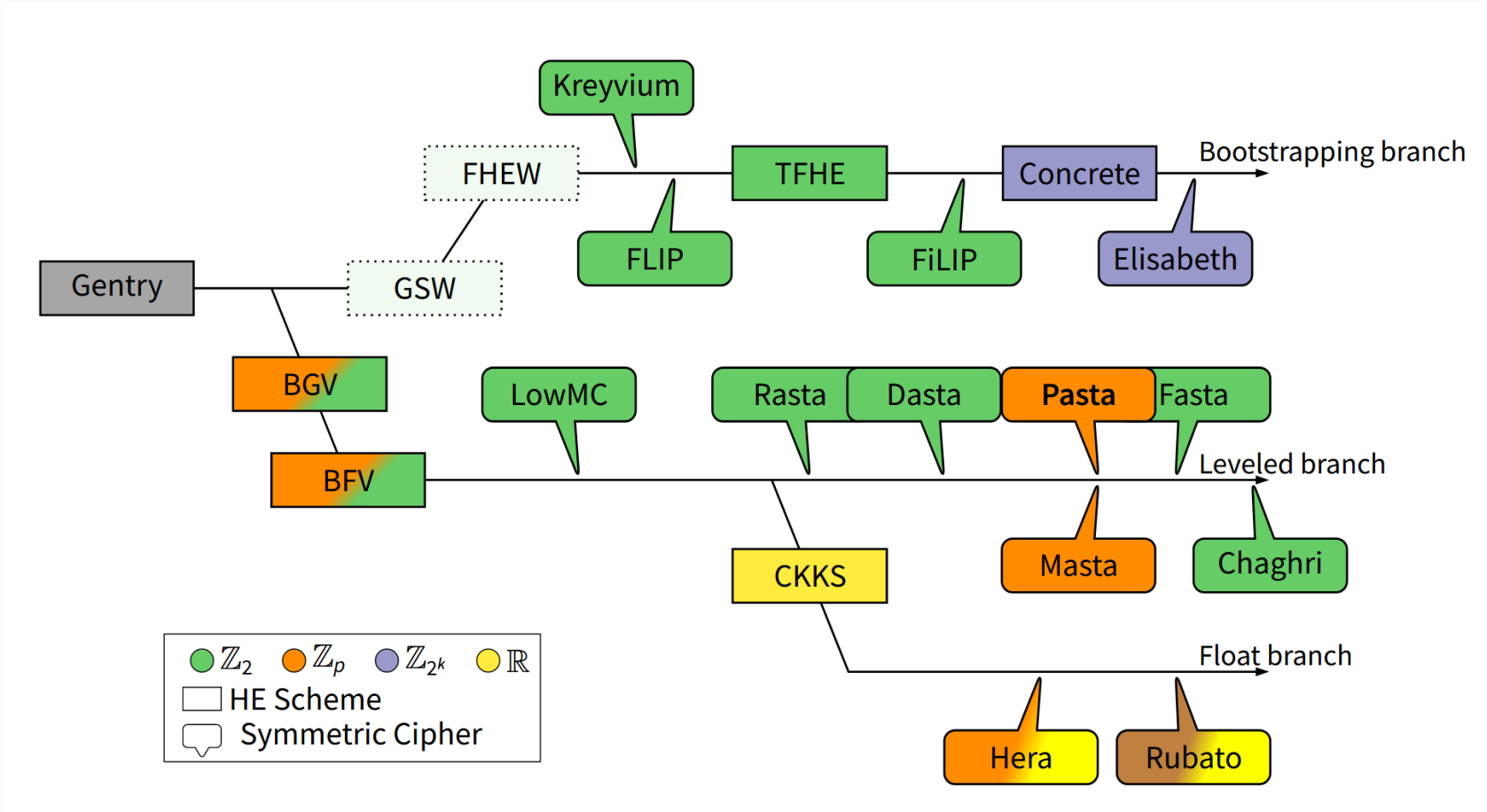
Role of symmetric-key crypto and hashing in systems



Transitions of use-cases in (symmetric) cryptography

- in the 1980s and 90s, there was a transition from hardware to software.
  - Hardware grew, but software grew much more.
- since the mid 2010s: we seem to be in a transition phase from direct implementations to indirect implementations within protocols aiming for "high functionality cryptography"
  - direct hardware and software implementations of course remain relevant, but the area of indirect implementations is growing fast.
  - new "virtual machines", new "metrics", co-developments of symmetric crypto with "higher/more functional" crypto layers

A Zoo of Ciphers for Hybrid Homomorphic Encryption, a.k.a. Transciphering



The ZK-friendly Hash Function Zoo

Type 1

"low degree only"

- Low-degree

$$y = x^d$$

- Fast in Plain
- Many rounds
- Often more constraints
- MiMC(16), GMiMC(19), Poseidon(19), Neptune (21), Poseidon2 (23)

The ZK-friendly Hash Function Zoo

Type 1

"low degree only"

- Low-degree

$$y = x^d$$

- Fast in Plain
- Many rounds
- Often more constraints
- MiMC(16), GMiMC(19), Poseidon(19), Neptune (21), Poseidon2 (23)

Type 2

"non-procedural", "fluid"

- Low-degree equivalence

$$y = x^{1/d} \Rightarrow x = y^d$$

- Slow in Plain
- Fewer rounds
- Fewer constraints
- Friday(18), Vision (19), *Rescue*(19), Grendel(21), Griffin (22), Anemoi (22)

The ZK-friendly Hash Function Zoo

Type 1

"low degree only"

- Low-degree

$$y = x^d$$

- Fast in Plain
- Many rounds
- Often more constraints
- MiMC(16), GMiMC(19), Poseidon(19), Neptune (21), Poseidon2 (23)

Type 2

"non-procedural", "fluid"

- Low-degree equivalence

$$y = x^{1/d} \Rightarrow x = y^d$$

- Slow in Plain
- Fewer rounds
- Fewer constraints
- Friday(18), Vision (19), *Rescue*(19), Grendel(21), Griffin (22), Anemoi (22)

Type 3

"lookups"

- Lookup tables

$$y = T[x]$$

- Very fast in Plain
- Even fewer rounds
- Constraints depend on proof system
- `Reinforced Concrete` (21), Tip5 (23), Tip4 (23), RC_p(23)

The MPC/Sharing-friendly Symmetric Crypto Zoo

2015: LowMC

2016: MiMC, LegendrePRF

2018: CryptoDarkMatter

2019: GMiMC

2020: HadesMiMC

2021: Ciminion, "CryptoDarkMatter++"

2022: Rain, AIM

2023: Hydra

Open-sourcing implementations: `zoos.iaik.tugraz.at`

- Hybrid HE Use-Case:
  - Extensive benchmarks in different HE libraries including use-cases
  - 16 implementations (various ciphers for various HE libraries), before the count was 1.
- MPC Use-Cases:
  - Implementations of MiMC, GMiMC, HadesMiMC, Rescue, Ciminion, Hydra
  - More elaborate framework allowing for various libraries, access structures, still to come
- Zero-Knowledge Use-Cases:
  - Zoo of plain implementations (8)
  - Proof knowledge of preimages of hash functions (6)
  - Proof membership witness in Merkle tree accumulators (6)

No, how did we get here?

An email from 2012... (1/3)

From: Christian Rechberger [mailto:c.rechberger@mat.dtu.dk]
Sent: Thursday, August 02, 2012 12:17 PM
To: Julia Borghoff; Anne Canteaut; Lars Ramkilde Knudsen; Gregor Leander
Subject: The cipher LowMC2

Hi all, for some time already I'm looking into ways of building ciphers with a small number of multiplications. A new design "LowMC2" is briefly outlined as follows:
An idea for a simple key-alternating cipher is to use the map x->x^3 with x being elements in GF(2^n) as a round function, and iterate it n times.

An email from 2012... (2/3)

This construction should have very good properties against vario linear or differential attacks, but an obvious attack vector are interpolation/algebraic attacks. However, with n rounds I'd think that as the degree is close to maximal, and almost all coefficients are present in a polynomial description of the output of the cipher. On the implementation side, even $2n$ rounds would be very competitive in the settings I have in mind, but of course it would be interesting to see with how little rounds one can get away with.

An email from 2012... (3/3)

A multiplication with always different random nxn matrices in GF(2), for use as a linear layer in the round transformation, but also for use as a way to derive round keys from the master key is another part of the design.

So, my question is: Any thoughts on such a construction? Would you be interested to work on this?
Best, Christian

2017: https://lowmc.github.io/fewmul "LeoCrypt v0"

FewMul 2017

Fewer Multiplications in Cryptography

Cryptographic primitives realized with few multiplications can significantly improve (or even enable!) applications in areas as diverse as homomorphic encryption, side-channel attack countermeasures, secure multiparty computation, or zero-knowledge proofs.

This one-time workshop aims to provide an overview of results, applications and current research in this area. This covers theory, design and analysis, as well as implementations. Major goals are to bring together researchers from the unusual set of relevant disciplines within cryptography/security and outside (e.g. circuit complexity), and to identify open problems and more applications.

- **Location:** Jussieu campus of Université Pierre et Marie Curie Room 116
- **Date:** Sunday, April 30th, 2017

2017: https://lowmc.github.io/fewmul "LeoCrypt v0"

Program

Time	Speaker	Title
08:00–09:00	Breakfast	
Intro and Session on Side-Channel Topics		
09:30–09:45	Christian Rechberger	Introduction
09:45–10:30	Begül Bilgin	FewMul-FewDepth-FewLength Triangle
10:30–11:00	Break	
Session on Foundations		
11:00–12:00	Rene Peralta	Functions with known multiplicative complexity
12:00–14:00	Lunch	
Session on Applications: MPC and SNARKS		
14:00–14:45	Arnab Roy	MiMC and SNARKS
14:45–15:30	Emmanuela Orsini	Efficient evaluation of symmetric primitives in MPC

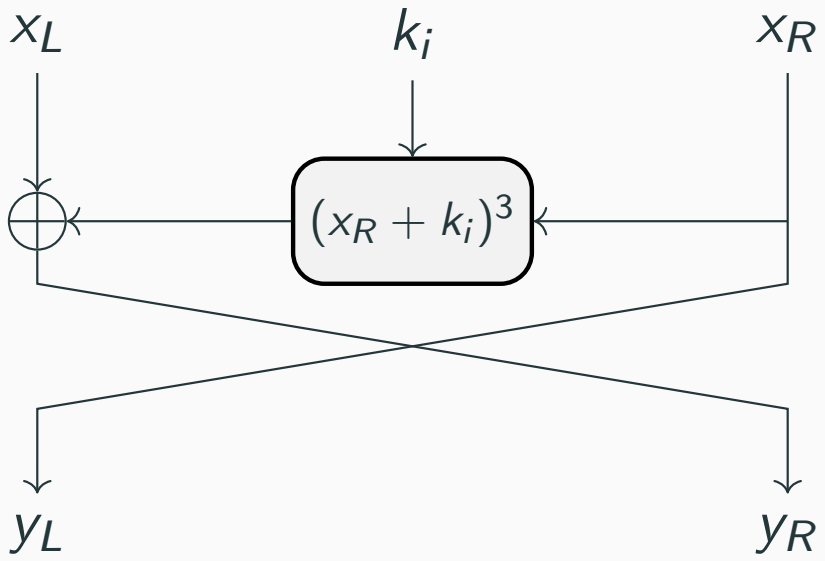
No really, how did we get here?

My sources for inspiration at that time:

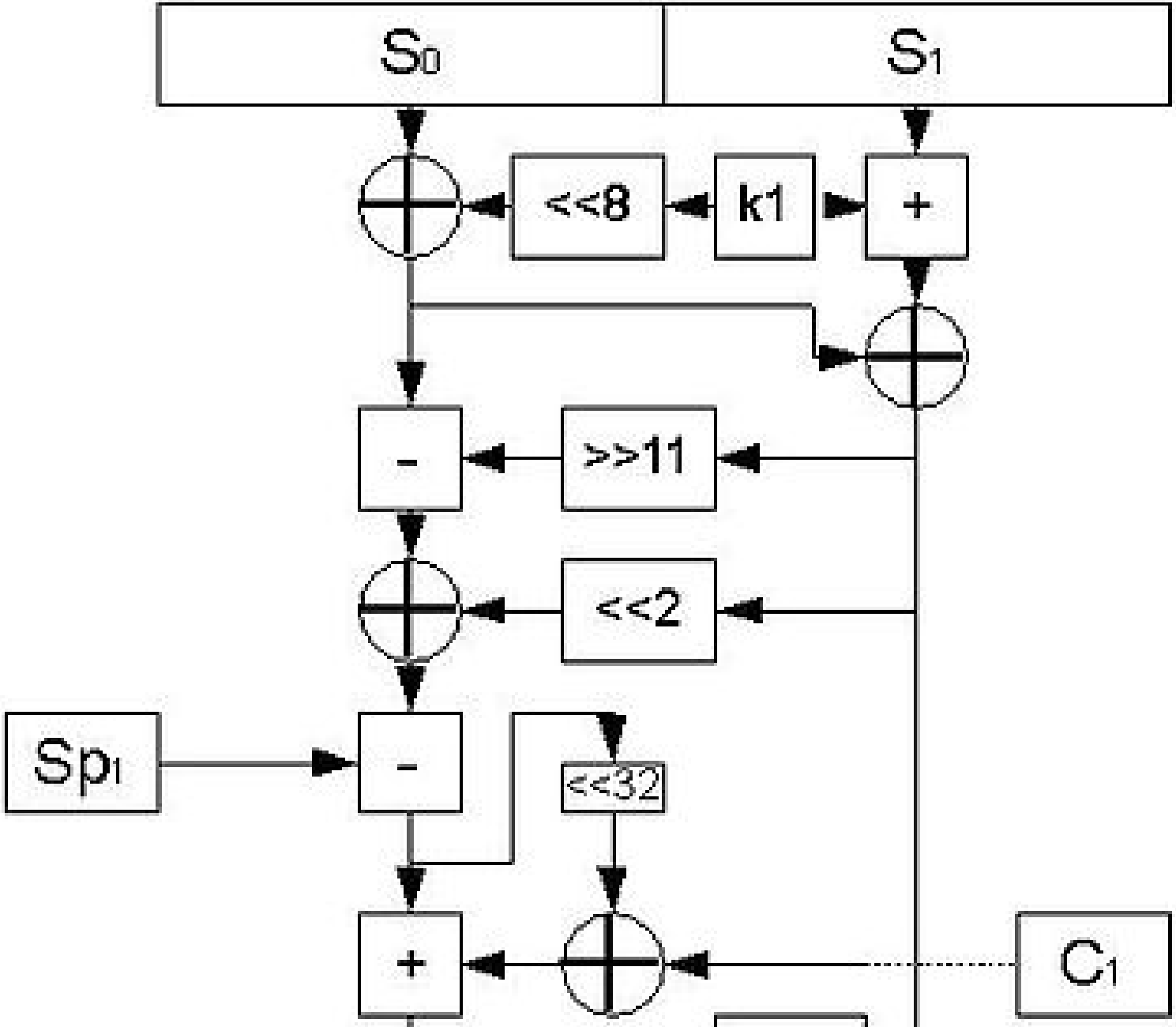
- 2009: "Secure Two-Party Computation is Practical" by Benny Pinkas, Thomas Schneider, Nigel P. Smart, and S. Williams
- 2011: "Can Homomorphic Encryption be Practical?" by Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan
- 2012: "On The Distribution of Linear Biases: Three Instructive Examples" by Mohamed Ahmed Abdelraheem, Martin Agren, Peter Beelen, and Gregor Leander
- 2014: "Zerocash: Decentralized Anonymous Payments from Bitcoin" by Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, Madars Virza

Prior art for (Feistel) MiMC

- $\mathcal{PURE}$ cipher [JK97] based on the $\mathcal{KN}$ Feistel cipher [NK95]



More prior art, for F(p) ciphers (1/2)



More prior art, for F(p) ciphers (2/2)

Richard Schroeppel: "The Hasty Pudding Cipher", submission to the NIST AES Competition, 1998.

First(?) F(p) cipher.

First tweakable block cipher

Flexible parameterization (blocksize, keysize), maybe a first too?

Ok. Where do we go from here?

On the "stability" of symmetric crypto and hashing

- MPC-friendly: Seems the most stable. Focus cryptanalysis efforts in standardization process/competition?
- HE-friendly: 4-5 underlying HE schemes are under standardization at ISO. Most, but not all schemes have a matching transciphering proposal.
- ZK-friendly: Most dynamic development at the moment.

In general, more cryptanalysis is definitely usefull and needed.

Underexplored directions?

- MPC-friendly hashing? Brought up by Luis Brandao in recent NIST call.
- On Hardware-friendly Sharing-friendly F(p) ciphers. Also relevant for cheap side-channel countermeasures.
  - Mathias Oberhuber: MiMC+ECC synergies in HW implementations. Both use multipliers same-size multiplier in in GF(2^n) or GF(p).
  - FX Standaert et al. AES-like F(p) ciphers!

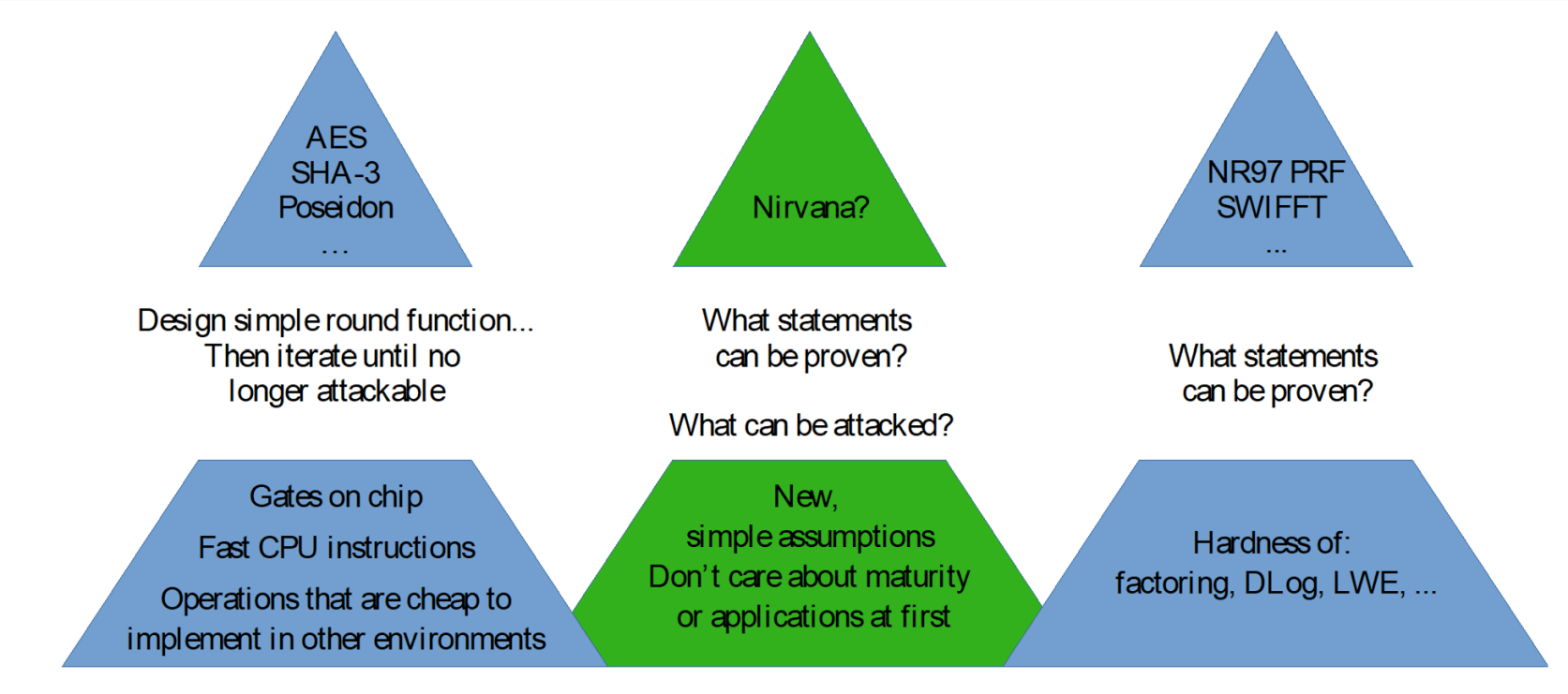
Classes of open problems

- How far can we go with signature schemes based symmetric crypto *only*? Signature size, computation effort?
- How far can we go with reducing computational overhead of hybrid homomorphic encryption?
- Holy grail in ZK-friendly hash function design: *Simultaneously* good performance in both plain and ZK
- Cryptanalysis of various new schemes in this domain

Thoughts on "Theory" vs. "Practice"

- Provable Security?
  - Modes of operation: do proofs carry over from F_2 to F_p?
  - SPN vs. Partial-SPN: First positive results by Guo, Standaert, Wang, Wang, Yu (FSE 22)
    - Stronger model, like indifferentiability?
- "Asymptotic analysis" / "asymptotic designs".
  - Input: blocksize, security level
  - Output: concrete design with security claim
  - Some designs allow for it, e.g. HPC, LowMC, MiMC, Poseidon, ...
  - Pros: Flexibility
  - Cons: Less focused cryptanalysis.

Thoughts on "Theory" vs. "Practice": A vision



Conclusions

- Lots of exciting new developments in "high functionality cryptography" - some are likely here to stay
- ... leading to lots of exciting research for design and analysis of symmetric crypto and hashing
- Industry interest is growing, demand for standards to support interoperability and increase trust

On the history of FHEMPCZK-friendly symmetric crypto

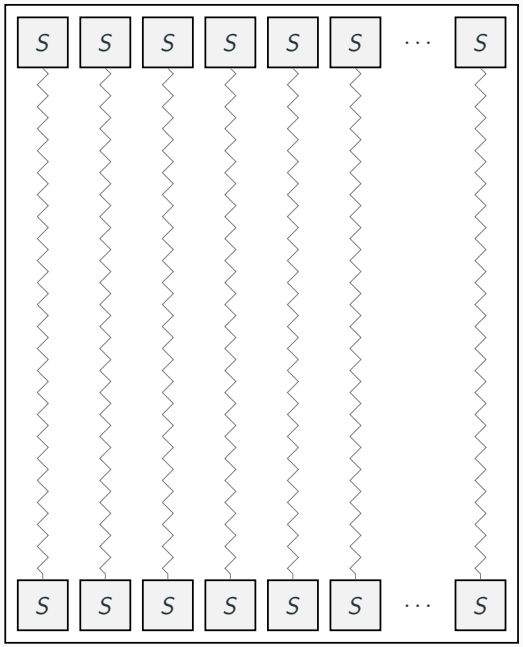
Christian Rechberger, TU Graz

LeoCrypt23

S-Box sizes, over time. A selection

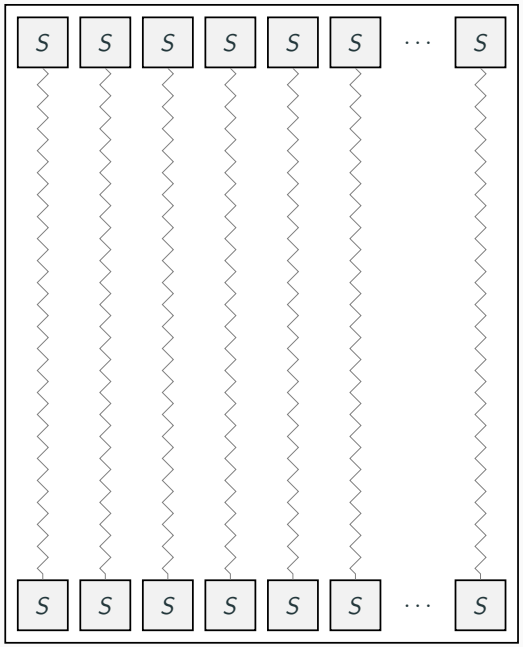
- mid 1970s, 6to4-bit: DES S-box just fits on a Chip
- mid 1990s, 8to8-bit: e.g. Rijndael/AES, attractive for good performance in both HW and SW
- since 2000, smaller, more "lightweight" S-boxes
  - 3to3-bit (e.g. Printcipher, LowMC)
  - 4to4-bit (e.g. Noekeon, Present, Klein, Prince)
  - 5to5-bit (e.g. Keccak, Ascon)
- since 2015, big and huge S-boxes
  - n to n-bit, elements in GF(2^n)
    - for n from 100 to 1000 (e.g. MiMC, Rain)
  - n to n-bit, elements in GF(p)
    - for n from 128 to $\geq$1000 (e.g. MiMC)
    - for n from 17 to 63 (e.g. Pasta)
    - for n from 8 to 128 (e.g. HadesMiMC, Poseidon, Marvellous)
    - for n from 8 to 128 (e.g. HadesMiMC, Poseidon, Marvellous)
  - set of size around 2^9 to 2^{10} to set of same size: (elements in $\mathbb{Z}_n$) ReinforcedConcrete (RChash)

⩓ SPNs with Partial Nonlinear Layers

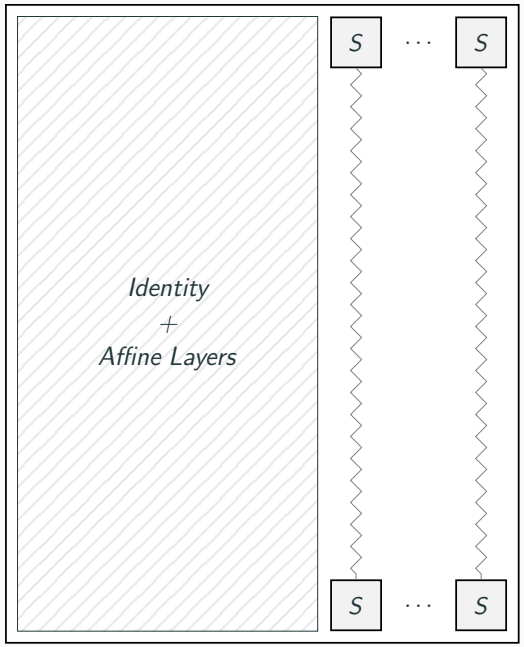


SPN
(e.g., SHARK in
1996)

⩓ SPNs with Partial Nonlinear Layers

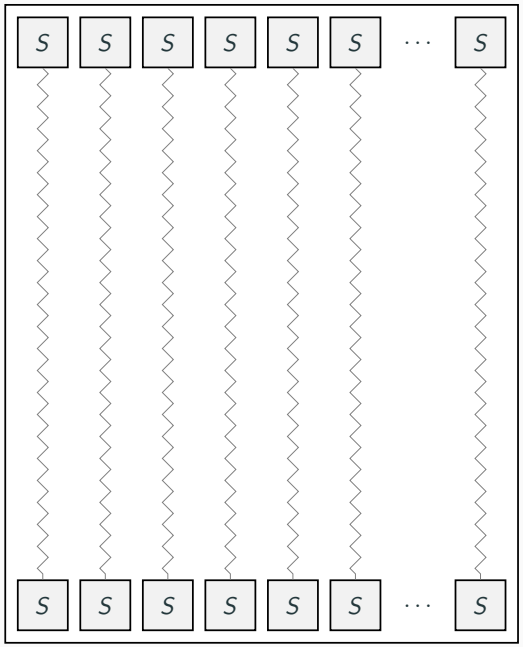


SPN
(e.g., SHARK in 1996)

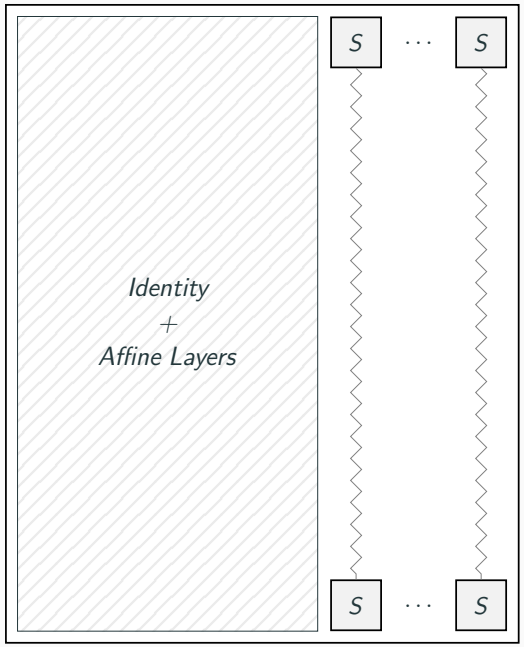


P-SPN since 2010
(e.g., ARMADILLO, Zorro, LowMC)

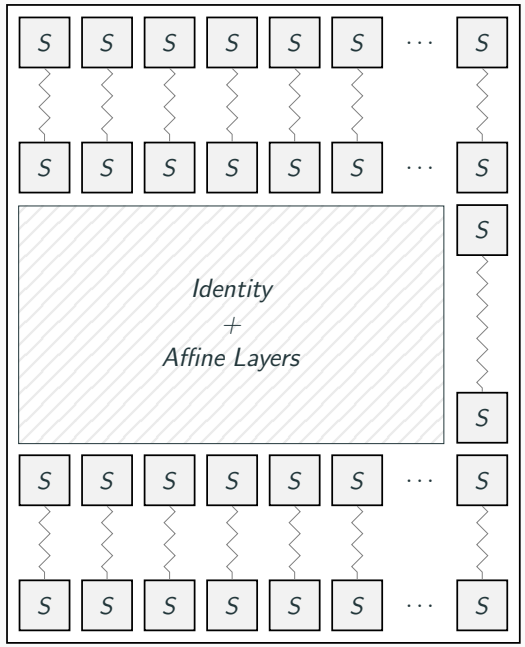
⩓ SPNs with Partial Nonlinear Layers



SPN
(e.g., SHARK in 1996)



P-SPN since 2010
(e.g., ARMADILLO, Zorro, LowMC)



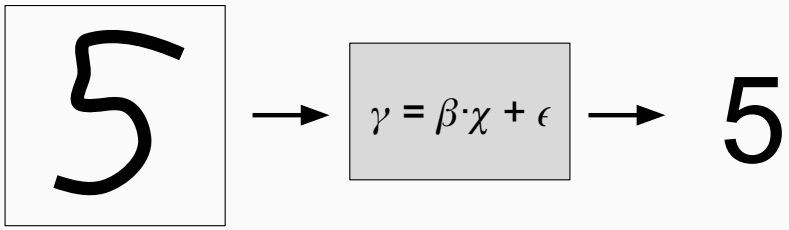
HADES since 2019
(e.g., HADESMiMC, POSEIDON, Ciminion, Hydra, Poseidon2)

Example: Hybrid Homomorphic encryption

Based on: "**Pasta – A Case for Hybrid Homomorphic Encryption**" by Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, Roman Walch

Cloud Computing

- Era of cloud computing:
  - *Outsource computation* to more powerful server
  - Server calculates *statistics*
  - *Machine learning as a service*
    - Use pre-trained, server-side ML models for classification
    - . . . Linear regression
    - . . . Deep neural networks

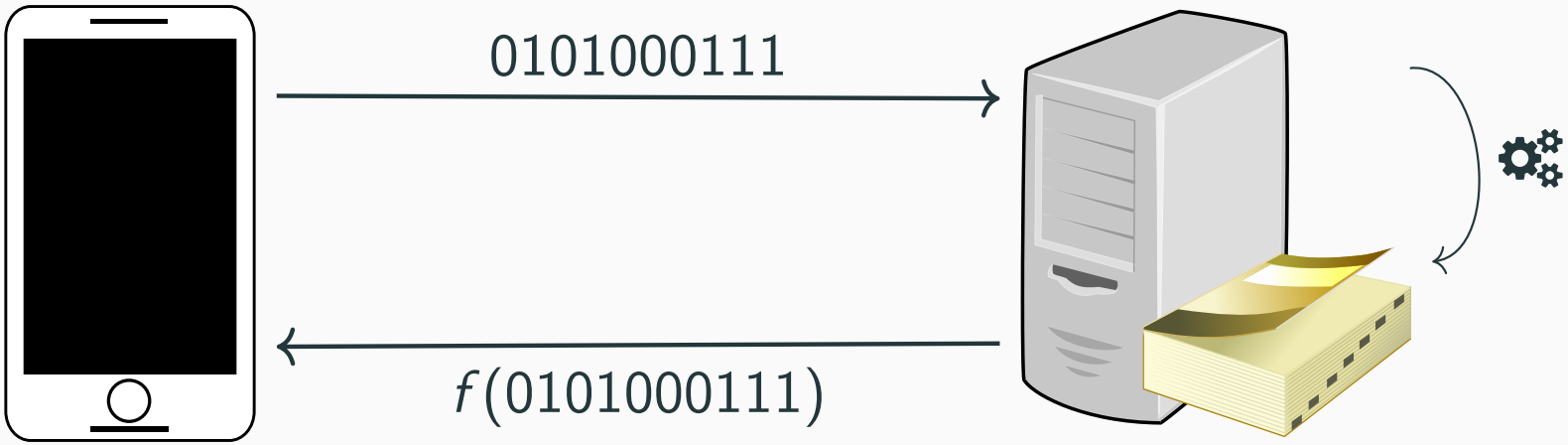


Outsourcing Computation

- Simple outsourcing

Data: 0101000111

Function: $f()$



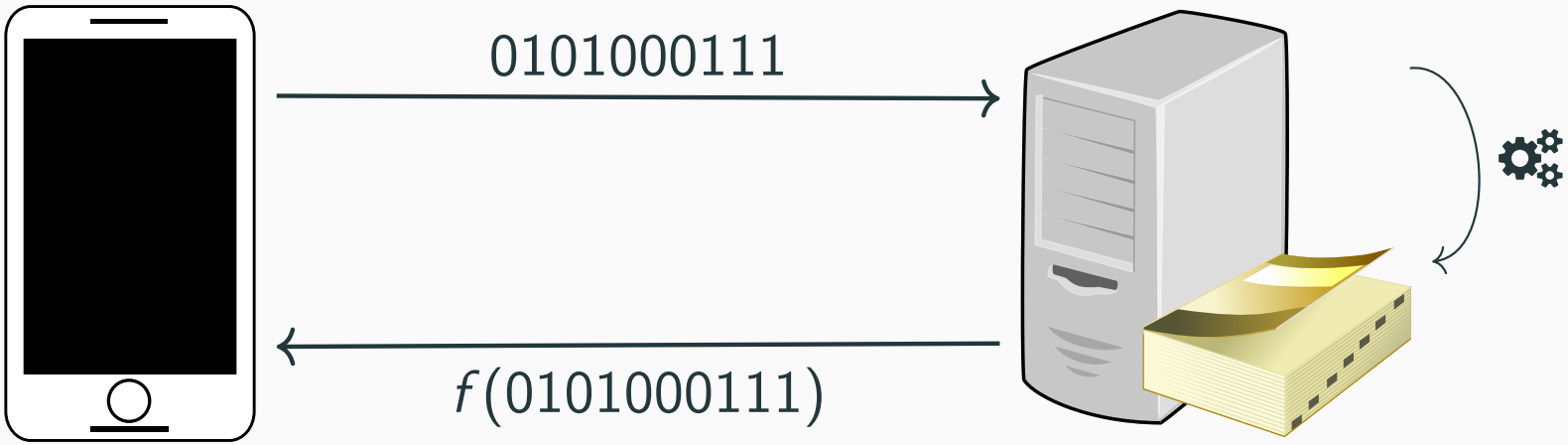
Outsourcing Computation

- Simple outsourcing

Data: 0101000111

Reveals Data

Function: $f()$



Homomorphic Encryption

- Operate on *encrypted*, unknown Data
- *Without knowing* secret decryption key

Homomorphic Encryption

- Operate on *encrypted*, unknown Data
- *Without knowing* secret decryption key

- Schemes and Libraries:
  - *BGV* [BGV12]: Integers, implemented in *HElib* [HS20]
  - *BFV* [Bra12; FV12]: Integers, implemented in *SEAL* [**sealcrypto**]
  - Problem: *Noise* in ciphertexts!
    - Metric: *multiplicative depth*

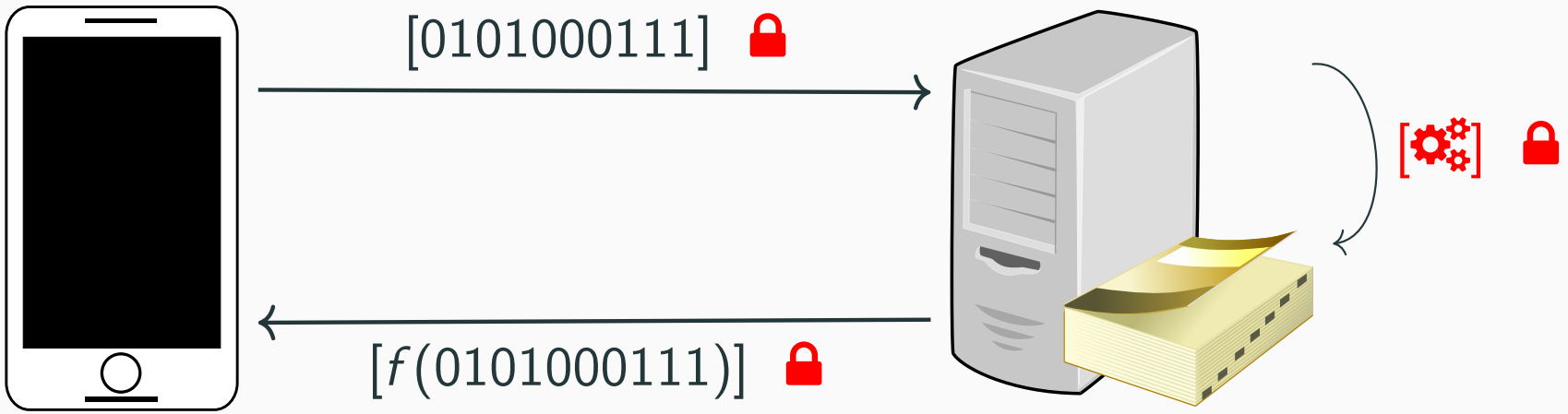
Outsourcing Computation - Homomorphic Encryption

- Simple outsourcing reveals data
  - $\Rightarrow$ Homomorphic encryption

HE Key:

Data: 0101000111

Function: $f()$



Outsourcing Computation - Homomorphic Encryption

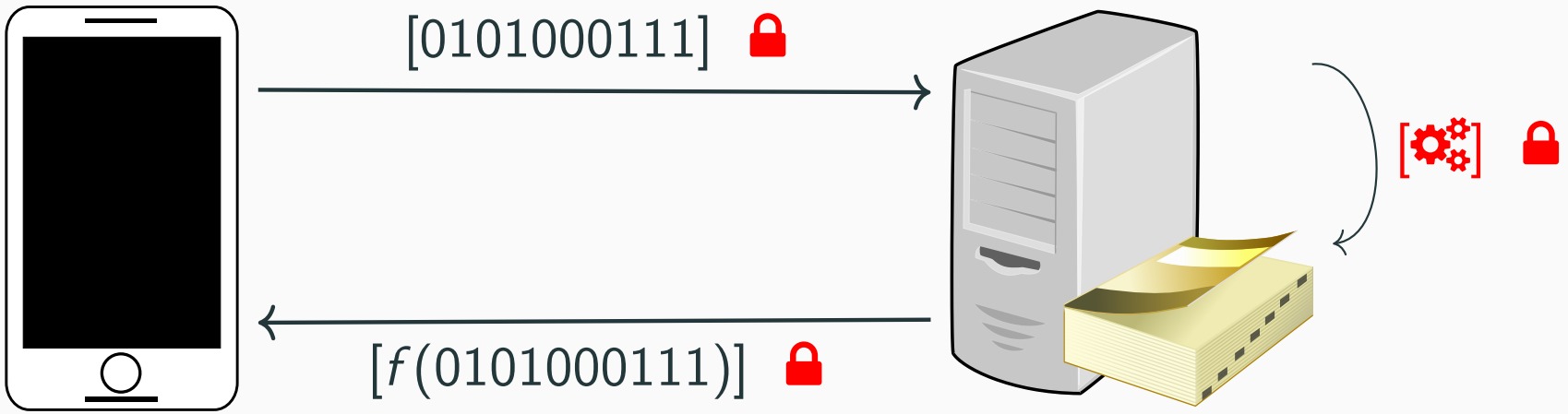
- Simple outsourcing reveals data
  - $\Rightarrow$ Homomorphic encryption

HE Key:

Huge Ciphertext Expansion

Data: 0101000111

Function: $f()$



Hybrid Homomorphic Encryption

- Homomorphic ciphertext are several orders of magnitude larger then plaintext
  - e.g., 7.4 MB for $\leq$ 250 kB of data (often significantly less data)

  $\Rightarrow$ Increased data transfer!
- Solution:
  - Send data encrypted using *symmetric ciphers*
    - Ciphertexts have same size as plaintexts
    - e.g., 250 kB for 250 kB of data
  - *Homomorphically decrypt data* before use case

$\Rightarrow$ No ciphertext expansion

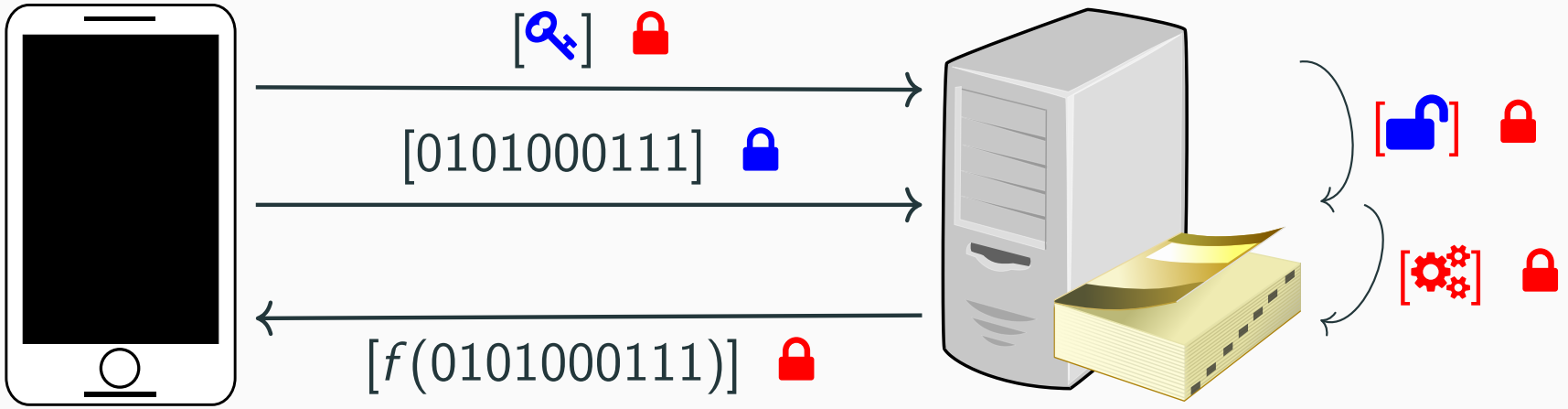
Outsourcing Computation - Hybrid HE

- Hybrid homomorphic encryption

HE Key: , Sym Key:

Data: 0101000111

Function: $f()$



Hybrid Homomorphic Encryption (cont.)

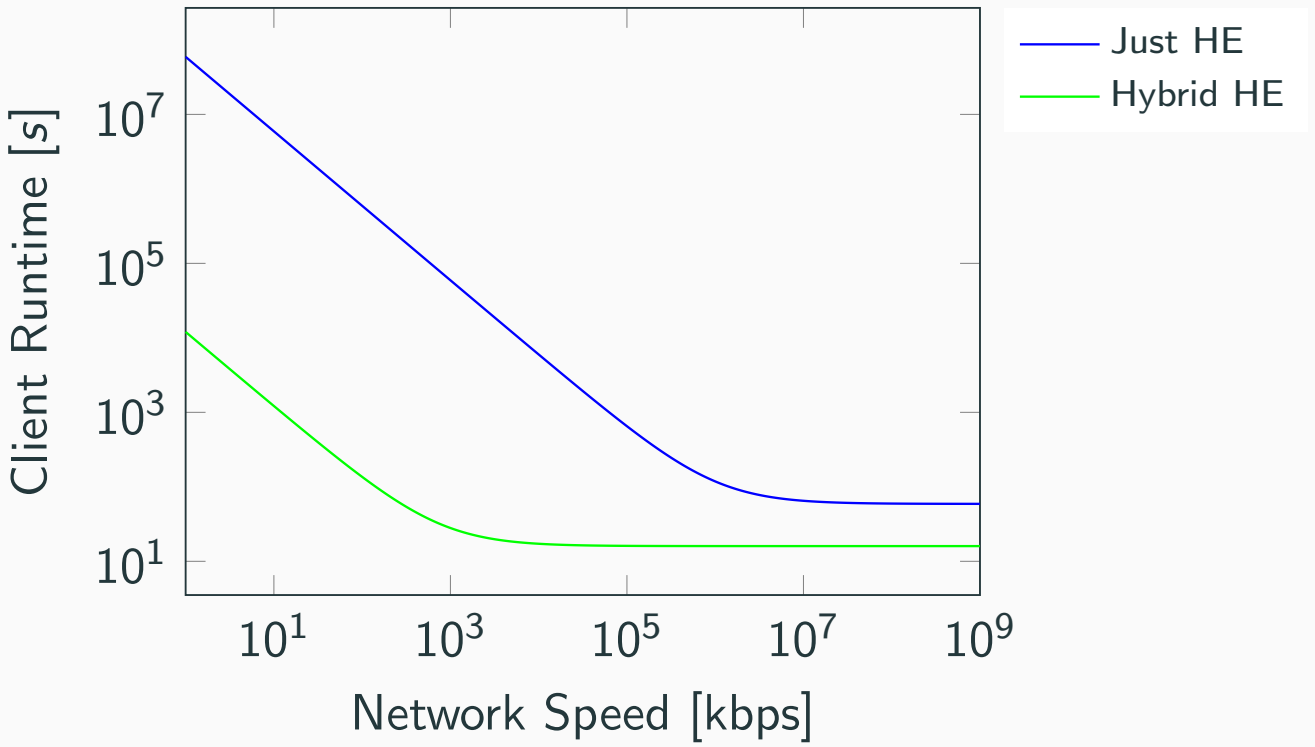
- Requires Sending of *homomorphically encrypted symmetric key*
  - Ciphertext expansion!
  - Amortized for large data
- Server evaluates *symmetric decryption circuit* before use case
  - Contributes to multiplicative depth of homomorphic computation
  - Requires shallow decryption circuit

$\Rightarrow$ Tradeoff: *data transmission vs. server runtime*

- Usable for constrained clients in slow networks!

Client Runtime Comparison

- Encryption + upload time depending on network speed:



Motivation

- Homomorphic encryption use case
- Different cost metric
  - *Depth* of the circuit important
  - Focus on low-round designs
- Avoid large classes of attacks in low-round construction
  - "Randomize" affine layers
- Efficient nonlinear layers for low depth?

Evolution of encryption schemes used for HHE

baseline 2012: AES: Efficient implementation of existing designs

new approach: few ANDs and low depth
2015: LowMC: Partial Sboxes, few ANDs, in an classical block cipher
2016: Kreyvium: Different tradeoffs from an improved steam cipher design

new: make relevant computations independent of the key
2016: FLIP: Offloading key register mixing
2018: Rasta: Offloading expensive affine layer generations
2020: Dasta: Make offloaded computation cheaper
2020: Masta: F(p) variant of Rasta, independent of this work

additional new idea: exploit structure of BFV and BGV schemes
2021: Pasta: focus on F(p), see later
2021: Fasta: F(2), speed-gains for concrete sizes

additional new idea: deal with noisy properties of CKKS
2021: Hera:

High-Level Take-Aways on HHE part

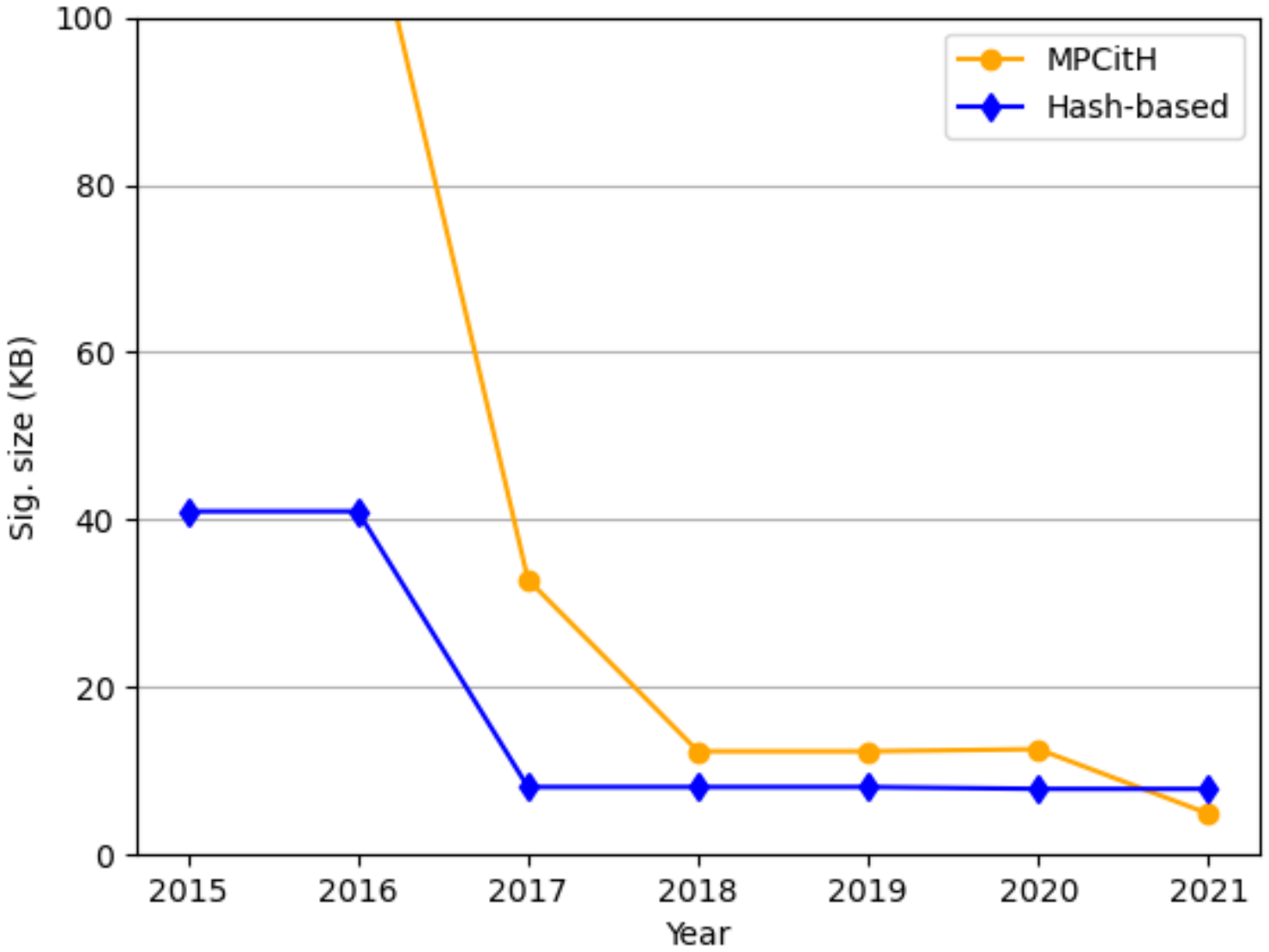
- Client Overhead:
  - *Bandwidth savings*:
    - *Only send actual data size* – Reduction by factor ≥ 20 ... Often significantly more!
  - *Runtime savings*:
    - Symmetric encryption $\geq 100\times$ faster as homomorphic encryption
- Server overhead:
  - *Addition to multiplicative depth:*
    - *Binary use case:* Agrasta: 4 cipher-cipher and 5 plain-cipher multiplications
    - *Integer use case:* Pasta: 4 cipher-cipher and 6 plain-cipher multiplications
  - *Runtime overhead:*
    - Depends on use case!
    - Large depth use cases: small factors ≤ 10
    - Small depth use cases: larger factors up to 1000

Example: MPC-in-the-head PK Signatures

or: how to go from signature sizes above 1MB to less than 5kB in case you only want to rely on symmetric crypto

Based on: "**Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto**" by
Christoph Dobraunig and Daniel Kales and Christian Rechberger and Markus Schofnegger and Greg Zaverucha

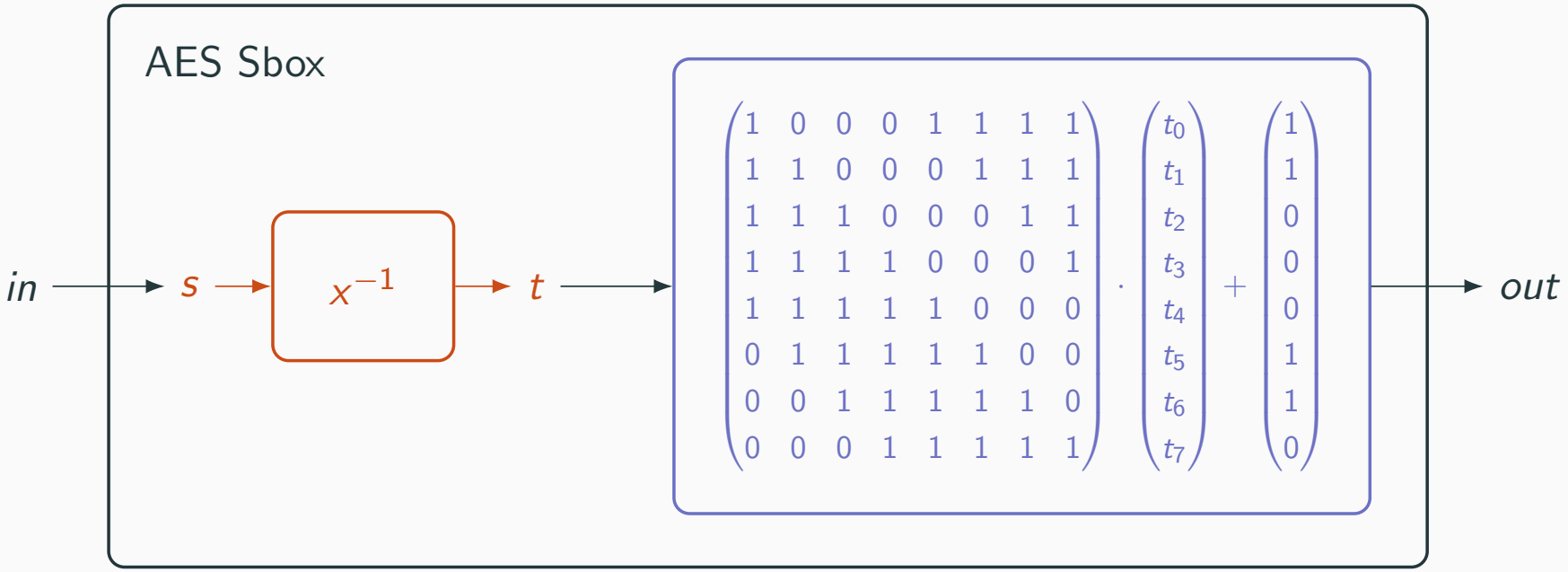
Recent Developments



The AES-Sbox

Inversion the only non-linear operation in AES

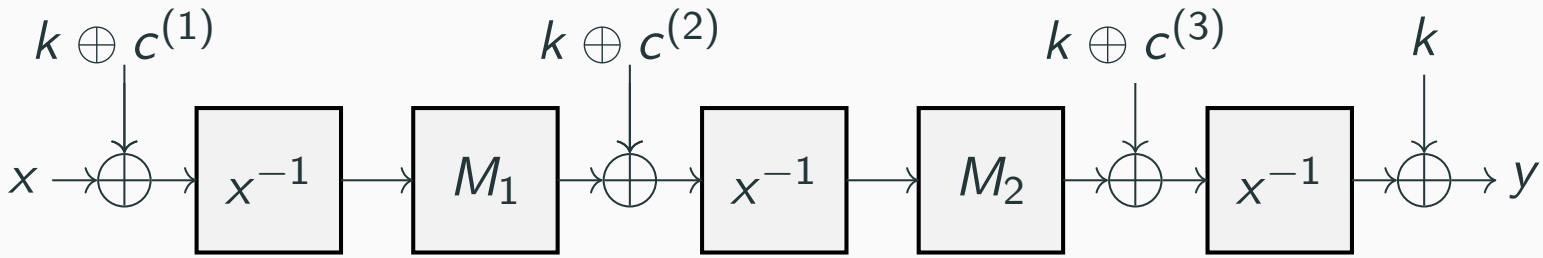
- Affine operations "free" in many MPC protocols (linear secret sharing)



Rain

Optimize one-way function for the specific use case:

- Large inverses over $\mathbb{F}_{2^n}$ for non-linearity
- Affine Transform without : Constant addition + multiplication with invertible matrix over $\mathbb{F}_2$ for linear mixing
- That is it, no key schedule, very few rounds



Signature schemes in the PQC Competition

Scheme	|pk|	Sig. size	Sign	Verify
Picnic1-L1-FS	32	32 860	1.60	1.31
Picnic3-L1	32	12 468	5.27	3.99
sphincss128sha256simple	32	8 080	248.37	0.75
sphincsf128sha256simple	32	16 976	14.73	1.79
Dilithium2	1 312	2 420	0.07	0.03
Falcon-512	897	666	0.11	0.02
Rainbow Ia-Classic	161 600	66	0.02	0.01
GeMSS128v2	352 188	33	320.99	0.08
Banquet-AES-128	32	19 776	7.03	5.32
Banquet-AES-128	32	13 284	47.31	43.03
Rainier$_3$-128 ($N = 16, \tau = 33$)	32	8 544	0.87	0.81
Rainier$_3$-128 ($N = 107, \tau = 20$)	32	6 176	2.96	2.92
Rainier$_3$-128 ($N = 1624, \tau = 14$)	32	4 880	28.28	28.16
Rainier$_3$-128 ($N = 7121, \tau = 11$)	32	4 496	105.98	105.15

Table 1: Comparison of public-key and signature sizes at the 128-bit security level for the third-round candidates of the NIST PQC standardization project and the designs explored in this work. Size in bytes, time in ms.

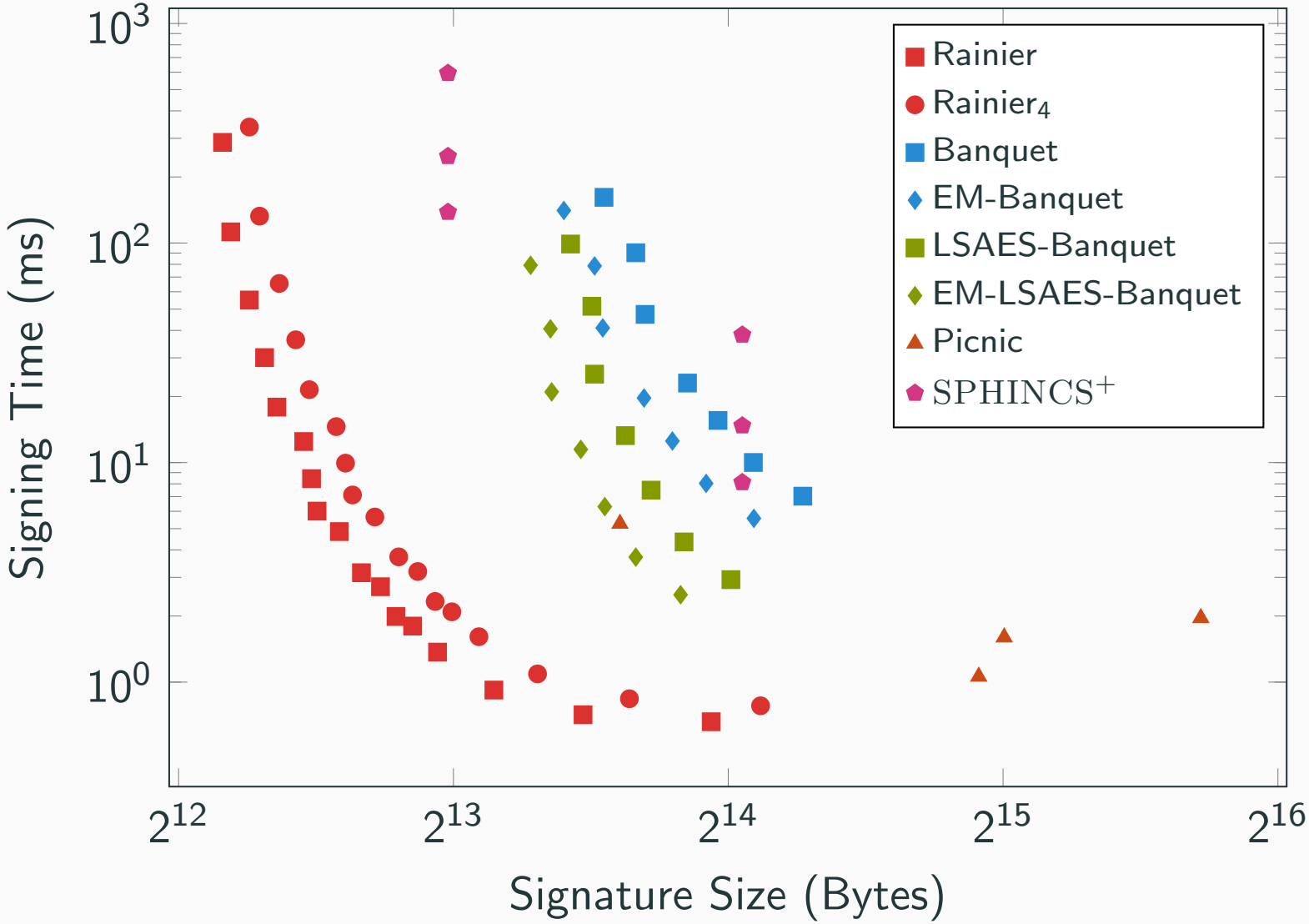
Security Level 128-bit pre-quantum, Timeline

Year	MPCitH + Standard	MPCitH + Custom
2016	ZKBoo, SHA-256: **1314.0 KB**	–
	ZKBoo, AES: **??? KB**	–
2017	ZKB++, AES: **209.0 KB**	Picnic1 ZKB++, LowMC: **32.9 KB**
2018	–	Picnic2 KKW, LowMC: **12.3 KB**
2019	BBQ, AES: **31.6 KB**	Picnic3 KKW, LowMC: **12.6 KB**
2021	Banquet, AES: **14.8 KB**	–
	RainS, AES: **10.8 KB**	RainS, Rain-3: **4.4 KB**

Security level 256-bit pre-quantum, timeline

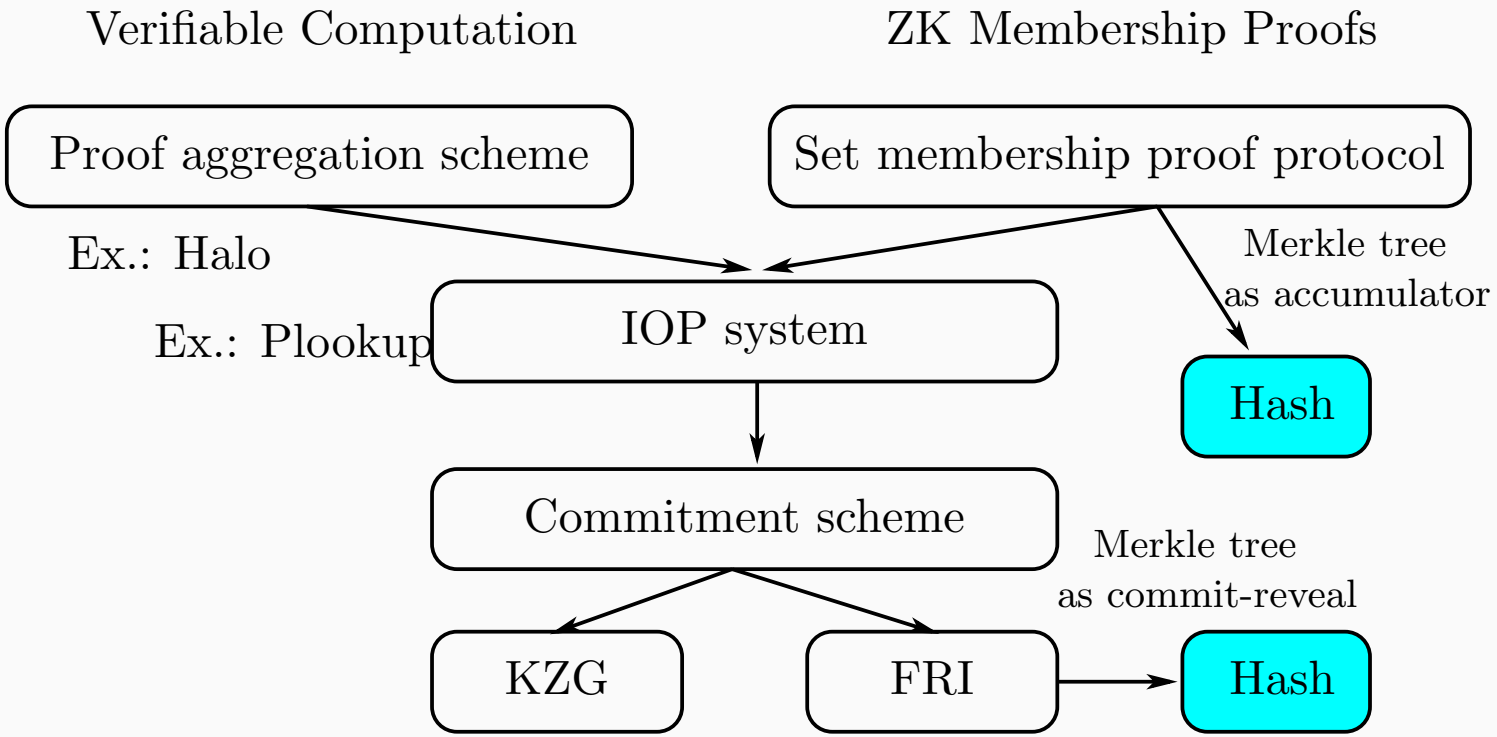
Year	Size
2017	Picnic1 ZKBoo++, LowMC: **128.2 KB**
2018	Picnic2 KKW, LowMC: **46.1 KB**
2019	Picnic3 KKW, LowMC: **48.7 KB**
2021	RainS, Rain-3: **20.0 KB**

Performance



Example: ZKP. Commitments schemes and set-membership proofs

Based on: "**Reinforced Concrete: Fast Hash Function for Zero Knowledge Proofs and Verifiable Computation**" by Mario Barbara, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, Roman Walch



Hash functions in set membership and verifiable computation protocols.

Open Problem

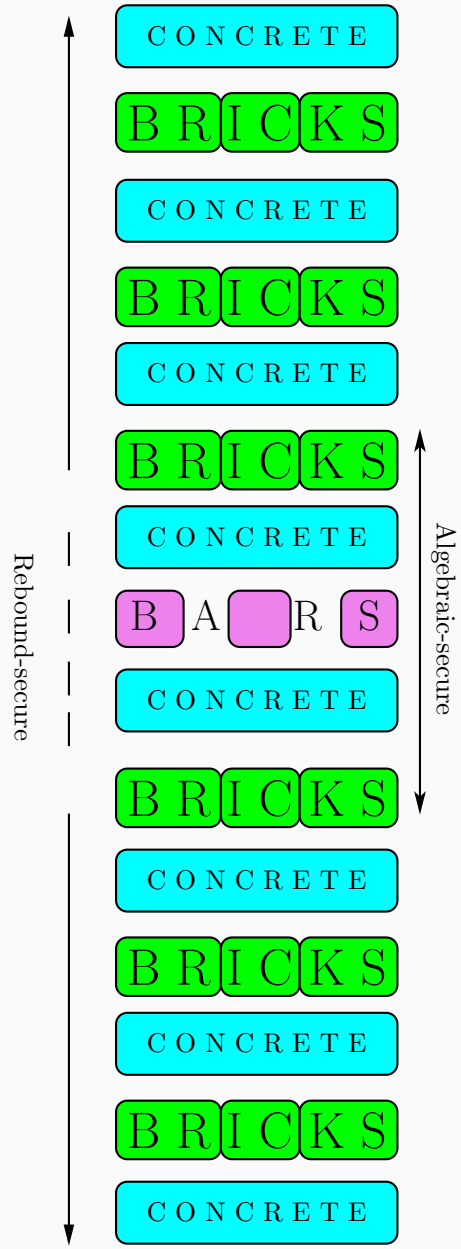
ZK-friendly hash function can't be nearly as fast as normal ones:

Hash	ZK time	x86 time	Cryptanalysis
Blake-2	100	1	10
Poseidon	1	100	1
Rescue	1	1000	1
Pedersen	4	500	100

We want:

- Faster hash on x86;
- Still ZK friendly
- Relies on long-term analysis.

Snapshot of Reinforced Concrete



Comparison with other Hash Functions

Name	Performance		
	Zero knowledge		Plain
	R1CS	Plookup	
	(gates)		(ms)
Poseidon-BLS/BN	243	438	19
Rescue-BLS/BN	288	364	415
Rescue-Prime-BLS/BN	252	321	362
Feistel-MiMC-BLS/BN	1326	1326	34
SHA-256	27534	$\approx$ 3000	0.37
Blake2s	21006	$\approx$ 2000	0.22
Sinsemilla	869*	670	131
`Reinforced Concrete-BN/BLS`	-	267	**3.3**
`Reinforced Concrete-FRI`	-	265	**1.03**

Table 2: Hashing 512 bits of data (two field elements) with different functions.

Cryptanalysis bounties/challenges/...

- Picnic/LowMC: Three rounds of challenges since 2020:
  - winners: Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, Hailun Yan, F. Betül Durak, Itai Dinur
  - `https://lowmcchallenge.github.io/`
- ZKProofs-friendly hashes, since 11/2021:
  - winners: Augustin Bariant, Clémence Bouvier, Gaëtan Leurent, Léo Perrin
  - `https://www.zkhashbounties.info/`

Algebraic Modelling of `Bar`

Our algebraic model of `Bar` is

$$\begin{cases} x = x_1 b_1 + x_2 b_2 + \cdots + x_n b_n \\ 0 = \prod_{k=0}^{s_i - 1} (x_i - k), \quad 1 \leq i \leq n \\ y_i = L_i(x_i), \quad 1 \leq i \leq n \\ y = y_1 b_1 + y_2 b_2 + \cdots + y_n b_n \end{cases}$$

with

$$b_i = \prod_{j>i} s_j = s_{i+1} s_{i+2} \ldots s_n,$$

and L_i = interpolation polynomial of S_i over $\mathbb{F}_p$

Idea behind `Bar`

- Ordinary base-b expansion: **fixed** base b

$$x = x_1 \cdot b^{n-1} + x_2 \cdot b^{n-2} + \ldots + x_{n-1} \cdot b + x_n \cdot 1$$

  with $0 \leq x_i < b$. E.g.

- Our expansion: **variable** base $s_1, \ldots, s_n$

$$x = x_1 \cdot s_2 s_3 \cdots s_n + x_2 \cdot s_3 s_4 \cdots s_n + \cdots + x_{n-1} \cdot s_n + x_n \cdot 1$$

  with $0 \leq x_i < s_i$.

- Variable base $\rightsquigarrow$ well-balanced decomposition of $p - 1$ (later)
- $s_1, \ldots, s_n$ different for each prime

Decomp: The Decomposition Function

The decomposition $\texttt{Decomp} : \mathbb{F}_p \to \mathbb{Z}_{s_1} \times \ldots \times \mathbb{Z}_{s_n}$ expands $x \in \mathbb{F}_p$ as

$$x = x_1 \cdot s_2 s_3 \cdots s_n + x_2 \cdot s_3 s_4 \cdots s_n + \cdots + x_{n-1} \cdot s_n + x_n$$

So we have: $x \longleftrightarrow (x_1, \ldots, x_n)$

SBox: The Centerpiece

- Let $(v_1, v_2, \ldots, v_n) := \texttt{Decomp}(p-1) \in \mathbb{Z}_{s_1} \times \ldots \times \mathbb{Z}_{s_n}$
- Take $p' :=$ next smaller prime of $\min_{1 \le i \le n} v_i$
- Then $\texttt{SBox} : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$ is defined as

$$y_i := S_i(x_i) := \begin{cases} f(x_i) & \text{if } x_i < p', \\ x_i & \text{if } x_i \ge p', \end{cases} \tag{1}$$

  where f denotes a permutation of $\mathbb{F}_{p'} = \mathbb{Z}_{p'}$
- f with maximal degree over $\mathbb{F}_{p'}$ (and dense representation)

Comp: The Composition Function

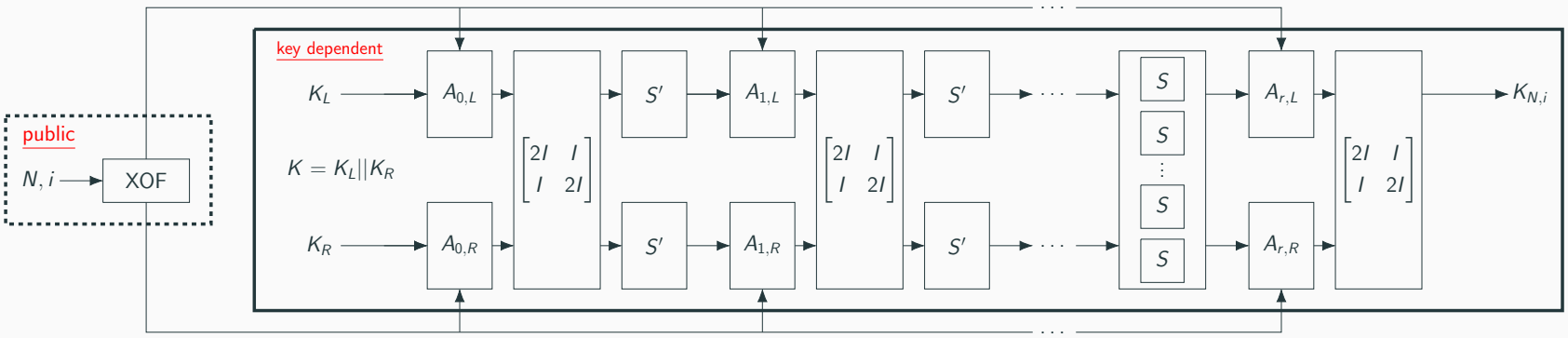
- $\texttt{Decomp} : x \mapsto (x_1, \ldots, x_n)$
- $\texttt{SBox} : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$
- $\texttt{Comp} : (y_1, \ldots, y_n) \mapsto y$
- Composition given by

$$y = y_1 \cdot s_2 s_3 \cdots s_n + y_2 \cdot s_3 s_4 \cdots s_n + \cdots + y_{n-1} \cdot s_n + y_n$$

- Overall we have

$$\texttt{Bar} = \texttt{Comp} \circ \texttt{SBox} \circ \texttt{Decomp}$$

The Pasta Design Strategy – Nonlinear Layer



- S-Boxes over $\mathbb{F}_p$ with $\lceil \log_2(p) \rceil \geq 16$
- Two layers S' and S
- We want to have a low overall degree
  - Depth directly impacts performance
- Lowest nonlinear degree is 2

  $\rightarrow$ Search for efficient degree-2 S-Box

The Pasta Design Strategy – Option 1: χ-Like S-Box

- Defined by
$$[\chi(\vec{x})]_i = x_i + x_{i+2} + x_{i+1}x_{i+2}$$
- In general no permutation over $\mathbb{F}_p^t$
- Used by e.g. Masta
- Efficient evaluation using rotations
$$\chi(\vec{x}) = \vec{x} + \mathtt{rot}_2(\vec{x}) \circ (\vec{1} + \mathtt{rot}_1(\vec{x})).$$

The Pasta Design Strategy – Option 2: Feistel-Like S-Box I

- Defined by

$$\begin{aligned} & S'(x_0 \| x_1 \| \cdots \| x_{s-1}) \\ = \; & x_0 \| (x_0)^2 + x_1 \| (x_1)^2 + x_2 \| \cdots \| (x_{s-2})^2 + x_{t-1}. \end{aligned}$$

- Can also be efficiently implemented using rotations

$$S'(\vec{x}) = \vec{x} + \left(\mathtt{rot}_{(-1)}(\vec{x}) \circ \vec{m} \right)^2,$$

- $\vec{m} = [0, 1, \ldots, 1]^T \in \mathbb{F}_p^t$ is a masking vector

The Pasta Design Strategy – Option 3: Feistel-Like S-Box II

- Defined by

$$\begin{aligned} & S''(x_0 \| x_1 \| \cdots \| x_{t-1}) \\ = \; & x_0 \| x_1 \| x_0 x_1 + x_2 \| x_1 x_2 + x_3 \| \cdots \| x_{t-3} x_{t-2} + x_{t-1}. \end{aligned}$$

- Can also be efficiently implemented using rotations

$$S''(\vec{x}) = \mathtt{rot}_{(-1)}(\vec{x}) \circ \mathtt{rot}_{(-2)}(\vec{x}) \circ \vec{m} + \vec{x}$$

- $\vec{m} = [0, 0, 1, \ldots, 1]^T \in \mathbb{F}_p^s$ is a masking vector

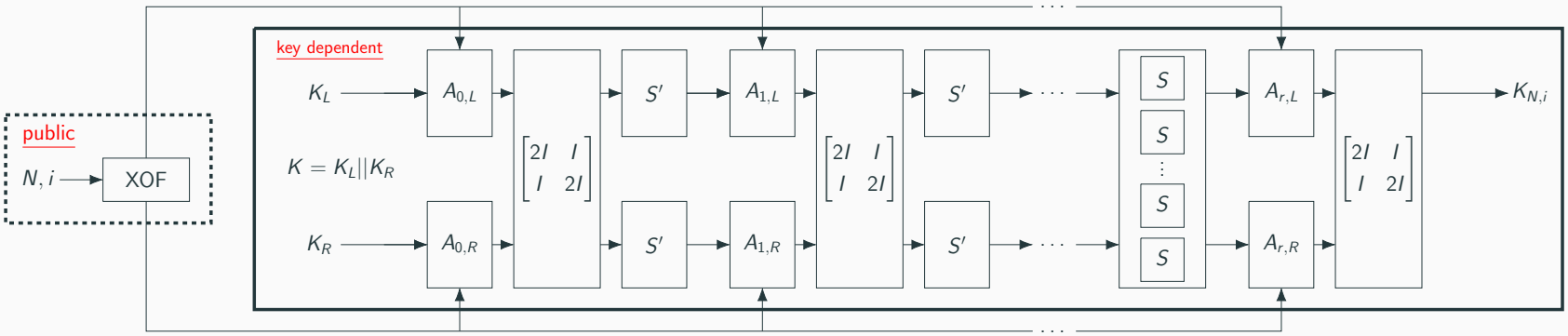
The Pasta Design Strategy – S-Box Comparison

- Compare homomorphic operations and multiplicative depth

S-box	pt-ct Add	ct-ct Add	pt-ct Mul	ct-ct Mul	Rot	pt-ct Depth	ct-ct Depth
χ	1	2	1	1	3	1	1
S'	-	**1**	**1**	**1**	**1**	**1**	**1**
S''	-	1	1	1	2	1	1

- We chose S' for efficiency

The Pasta Design Strategy – Degree-2 S-Box (Layer S')

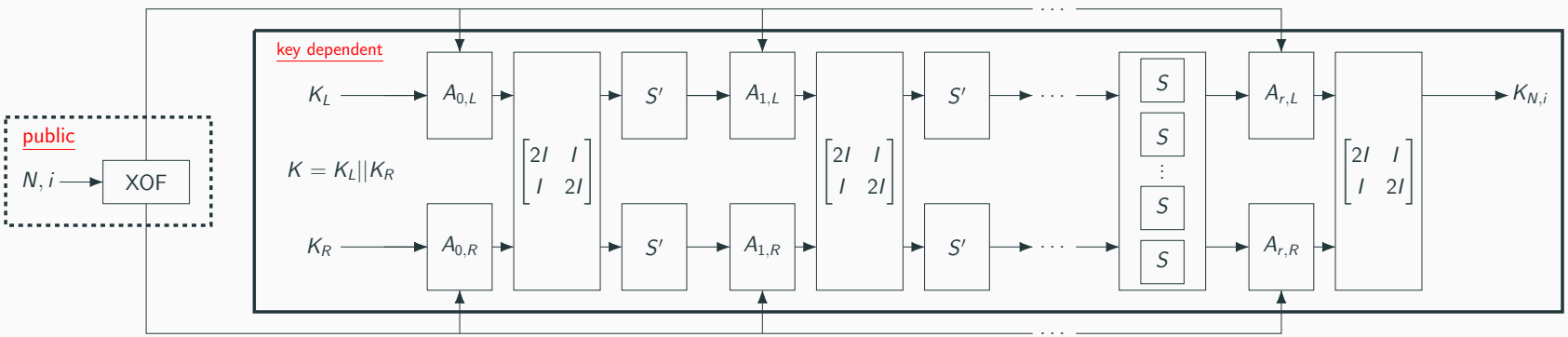


- Feistel-like S-Box S' defined by $[S'(\vec{x})]_0 = x_0$ and

$$[S'(\vec{x})]_i = x_i + (x_{i-1})^2 \quad \text{for} \quad i > 0$$

- Only one multiplication per word for $i > 0$
- Few issues for 3-round design (our goal)
  - Overall degree $2^3 = 8$ too weak against linearization (large state size)
  - Low-degree inverse (degree 2)

The Pasta Design Strategy – Degree-3 S-Box (Layer S)



- Simple degree-3 S-Box defined by $[S'(\vec{x})]_i = x_i^3$
- Bonus: High-degree inverse even without truncation
- Overall degree of $2^2 \cdot 3 = 12$ sufficient

Conclusion on Pasta HHE work

- Hybrid Homomorphic Encryption:
  - *Tradeoff: Client upload size vs server runtime*
- Majority of ciphers for HHE over $\mathbb{Z}_2$
  - Bad performance for integer use cases
- *Design of* Pasta
  - Optimized HHE cipher over $\mathbb{F}_p$
  - Paper: `https://eprint.iacr.org/2021/731.pdf`
- Extensive benchmarks in different HE libraries including use cases
  - Framework: `https://github.com/IAIK/hybrid-HE-framework`

Bibliography i

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "(Leveled) fully homomorphic encryption without bootstrapping". In: *ITCS*. ACM, 2012, pp. 309–325.

Zvika Brakerski. "Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP". In: *CRYPTO*. Vol. 7417. Lecture Notes in Computer Science. Springer, 2012, pp. 868–886.

Junfeng Fan and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption". In: *IACR Cryptol. ePrint Arch.* (2012), p. 144.

Bibliography ii

Shai Halevi and Victor Shoup. "Design and implementation of HElib: a homomorphic encryption library". In: *IACR Cryptol. ePrint Arch.* (2020), p. 1481.

Thomas Jakobsen and Lars R. Knudsen. "The Interpolation Attack on Block Ciphers". In: *FSE*. Vol. 1267. Lecture Notes in Computer Science. Springer, 1997, pp. 28–40.

Kaisa Nyberg and Lars R. Knudsen. "Provable Security Against a Differential Attack". In: *J. Cryptology* 8.1 (1995), pp. 27–37.