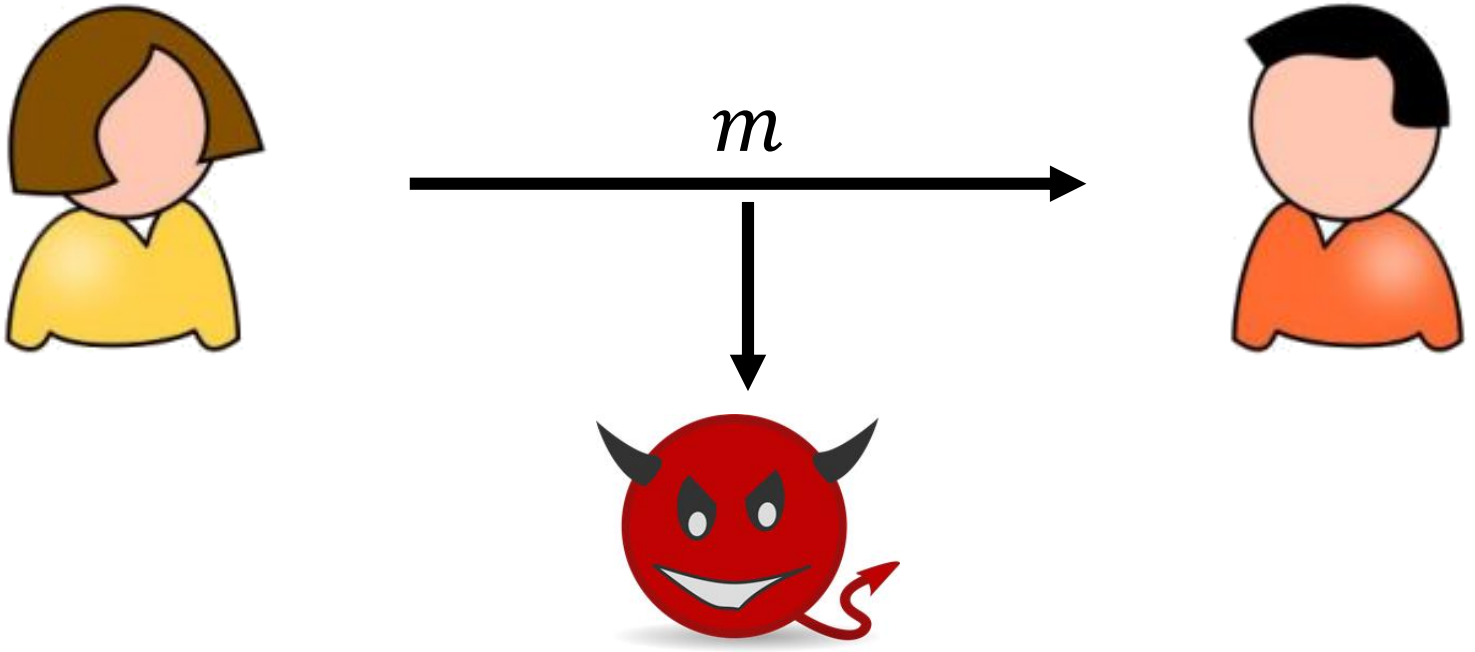
Security and Privacy through Modern Cryptography

David Wu

Stanford University

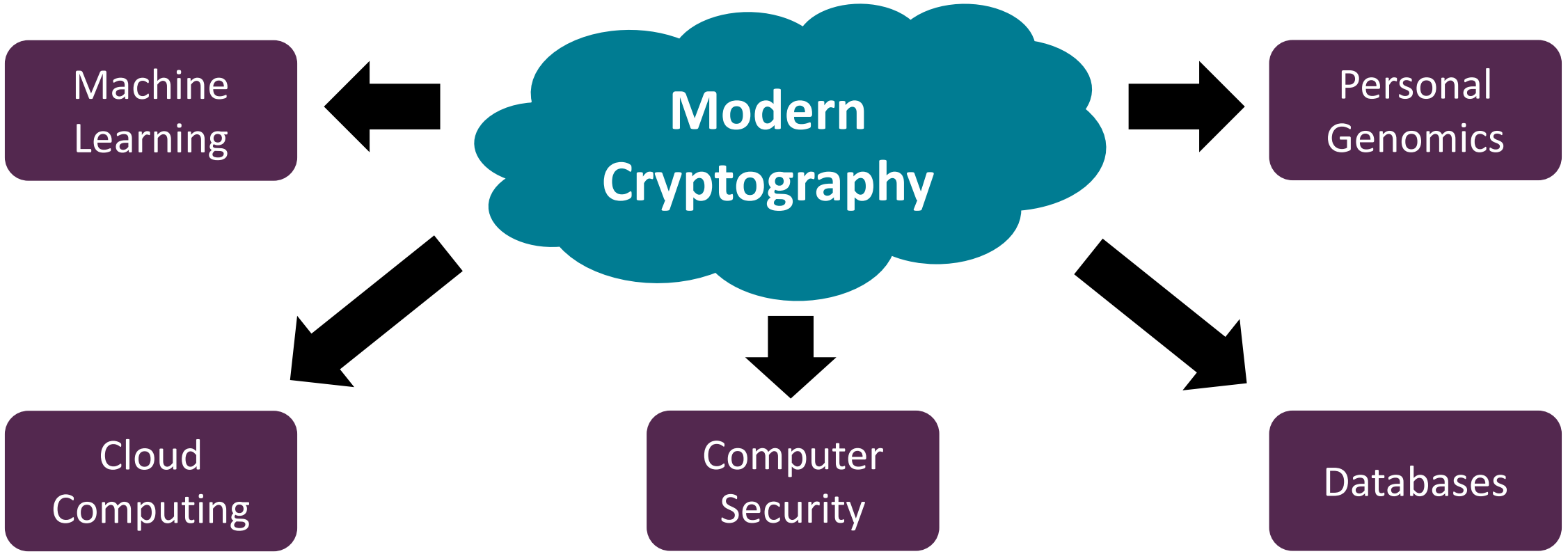
Cryptography in the 1970s

How can two users who have never met before communicate securely with each other?



secrecy integrity authenticity

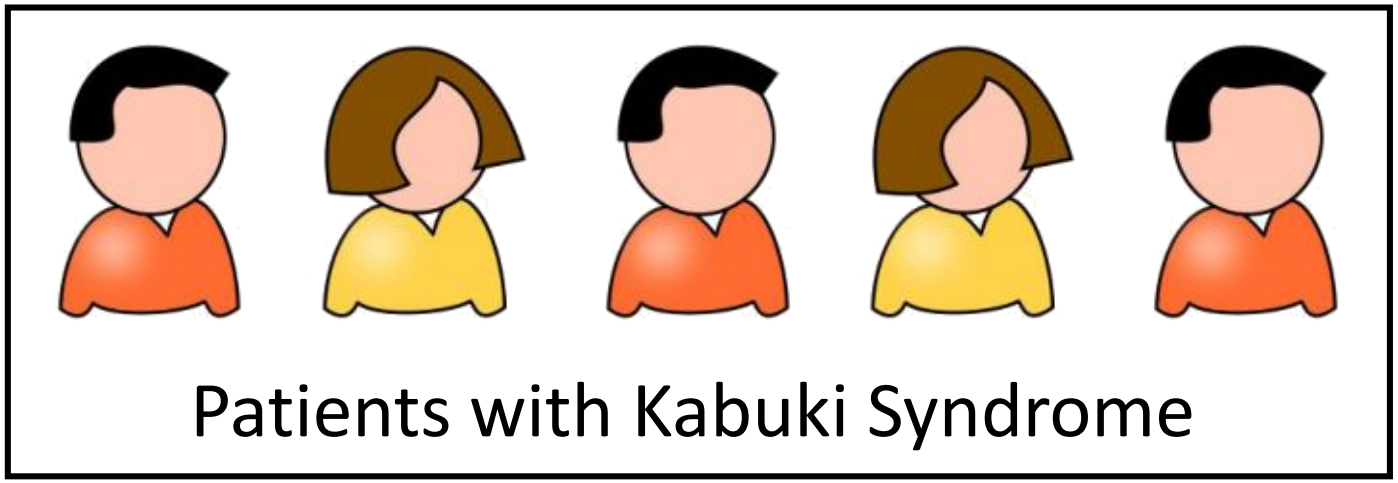
Modern Cryptography



Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh

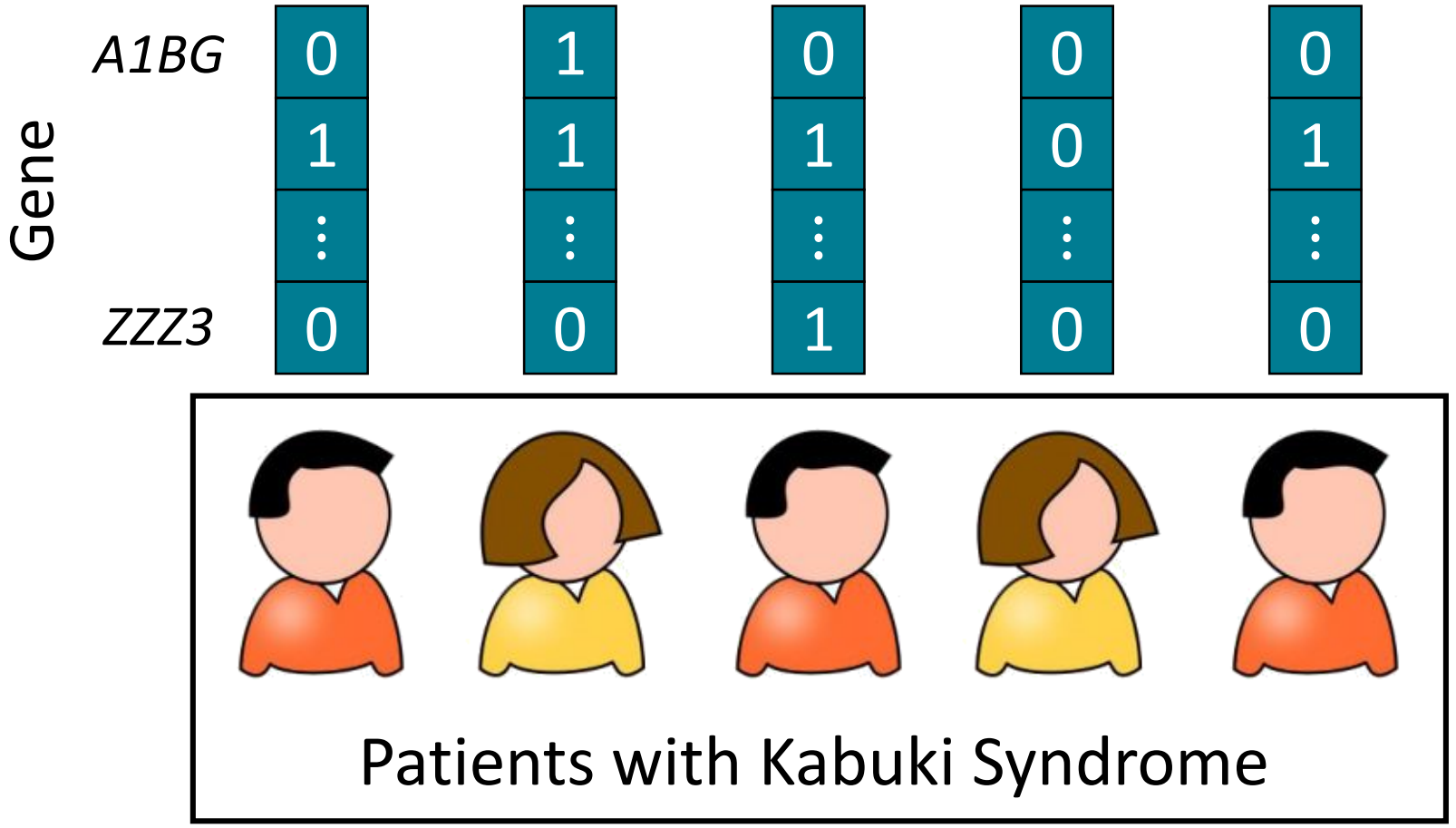
What gene causes a specific (rare) disease?



Each patient has a list of 200-400
rare variants over ≈20,000 genes

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh



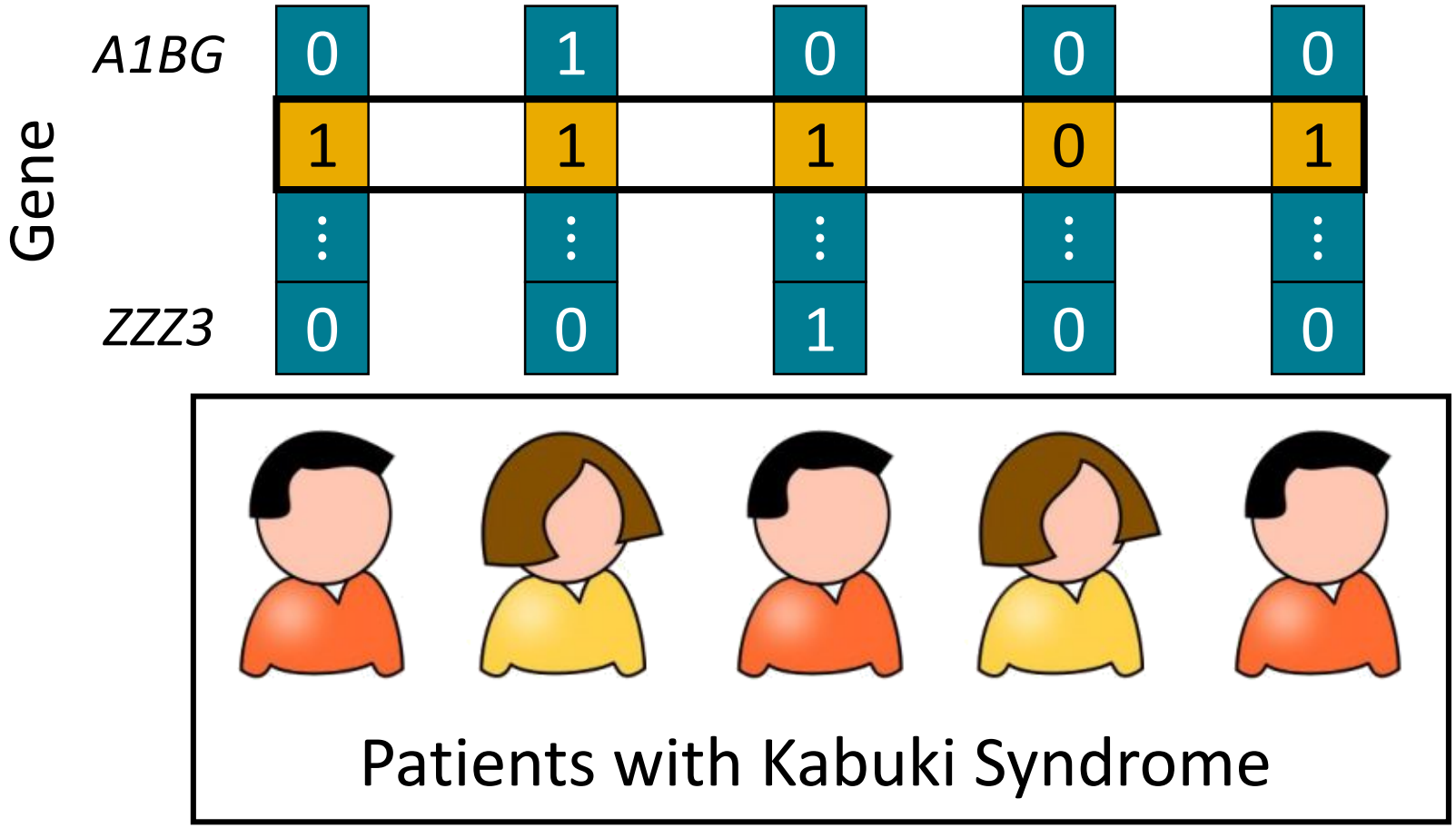
Each patient has a list of 200-400 rare variants over ≈20,000 genes

Each patient has a vector v where $v_i = 1$ if patient has a rare variant in gene i

Goal: Identify gene with most variants across all patients

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh



Each patient has a list of 200-400 rare variants over ≈20,000 genes

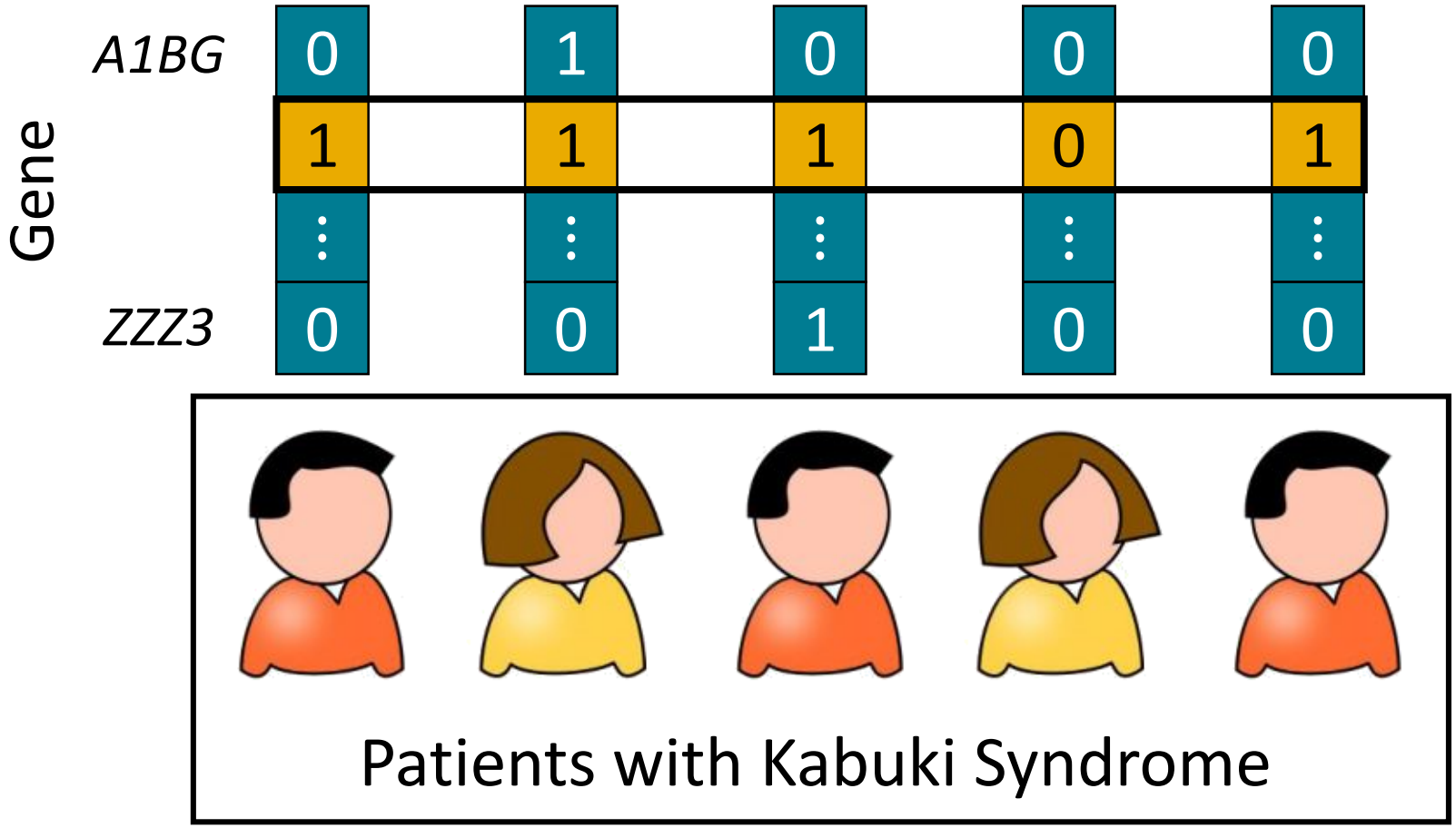
Each patient has a vector v where $v_i = 1$ if patient has a rare variant in gene i

Goal: Identify gene with most variants across all patients

Works well for monogenic diseases

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh



Each patient has a list of 200-400 rare variants over ≈20,000 genes

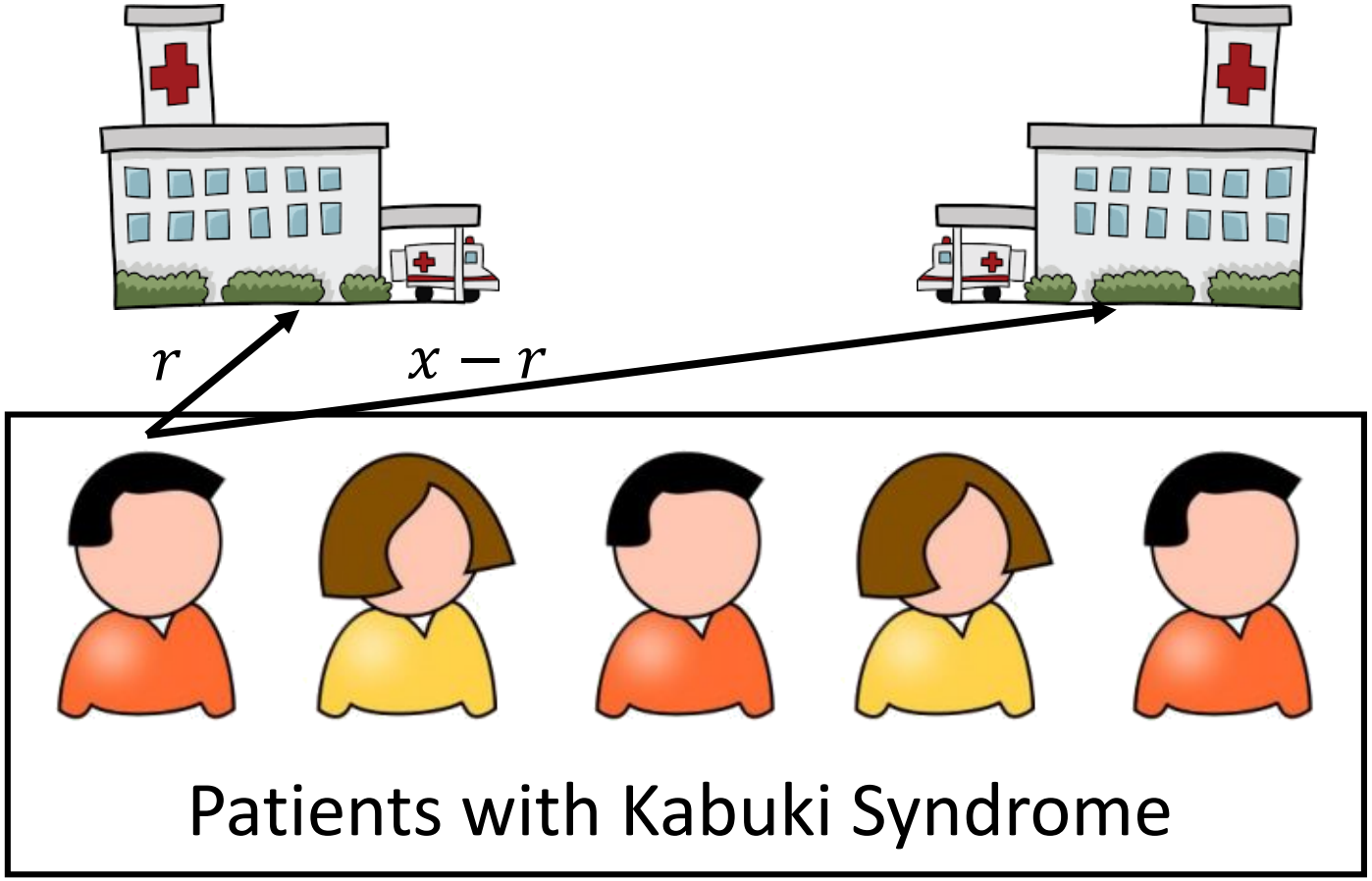
Question: Can we perform this computation without seeing complete patient genomes?

Goal: Identify gene with most variants across all patients

Works well for monogenic diseases

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh

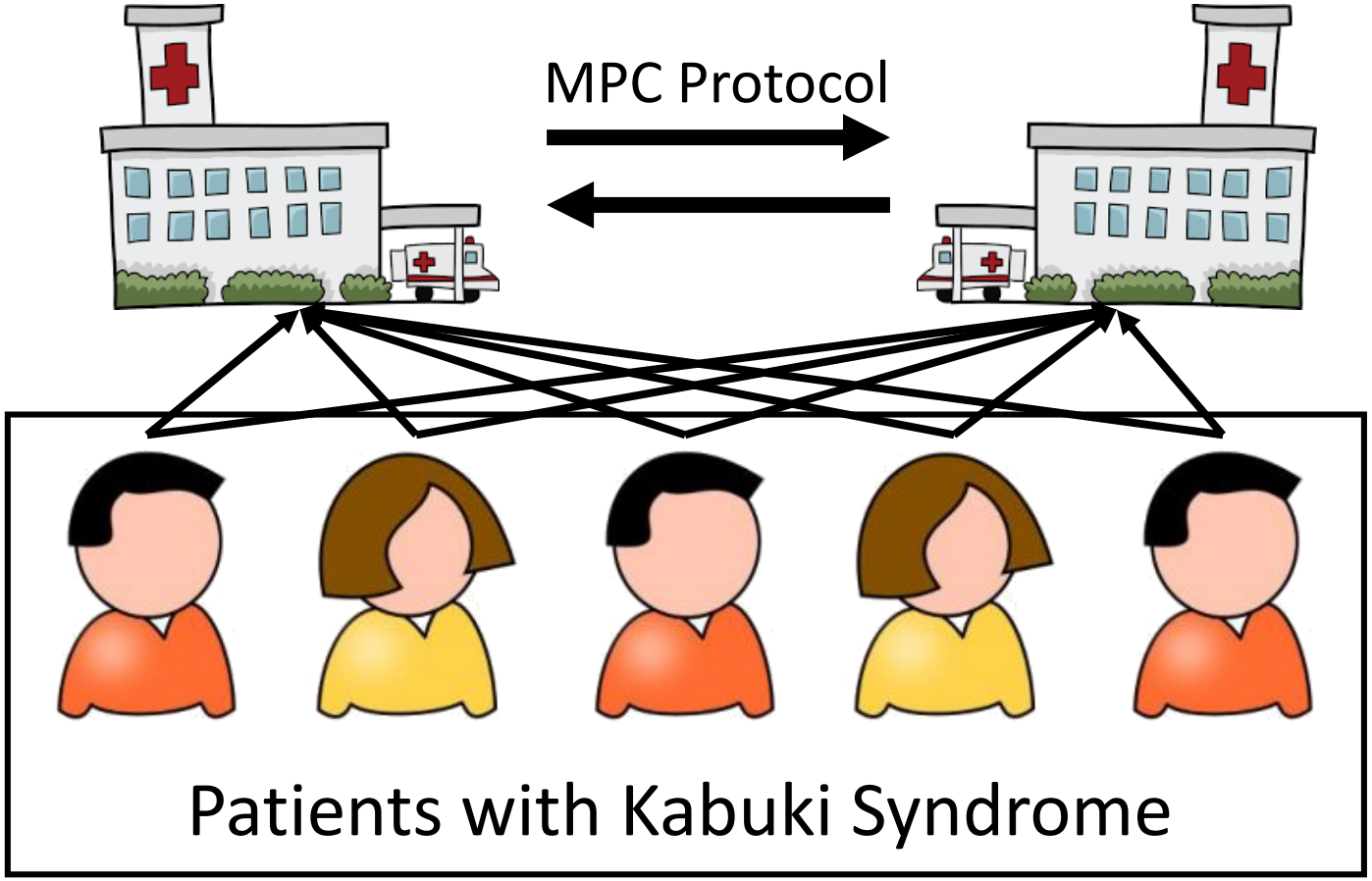


Each patient has a list of 200-400 rare variants over ≈20,000 genes

Patients "secret share" their data with two (non-colluding) hospitals

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh



Each patient has a list of 200-400 rare variants over ≈20,000 genes

Hospitals run a multiparty computation (MPC) protocol on pooled inputs

Patients "secret share" their data with two (non-colluding) hospitals

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh

MPC Protocol

Experiments on *real* patient data

Patients with Kabuki Syndrome

Each patient has a list of 200-400 rare variants over ≈20,000 genes

Top variants (sorted):
KMT2D, COL6A1, FLNB

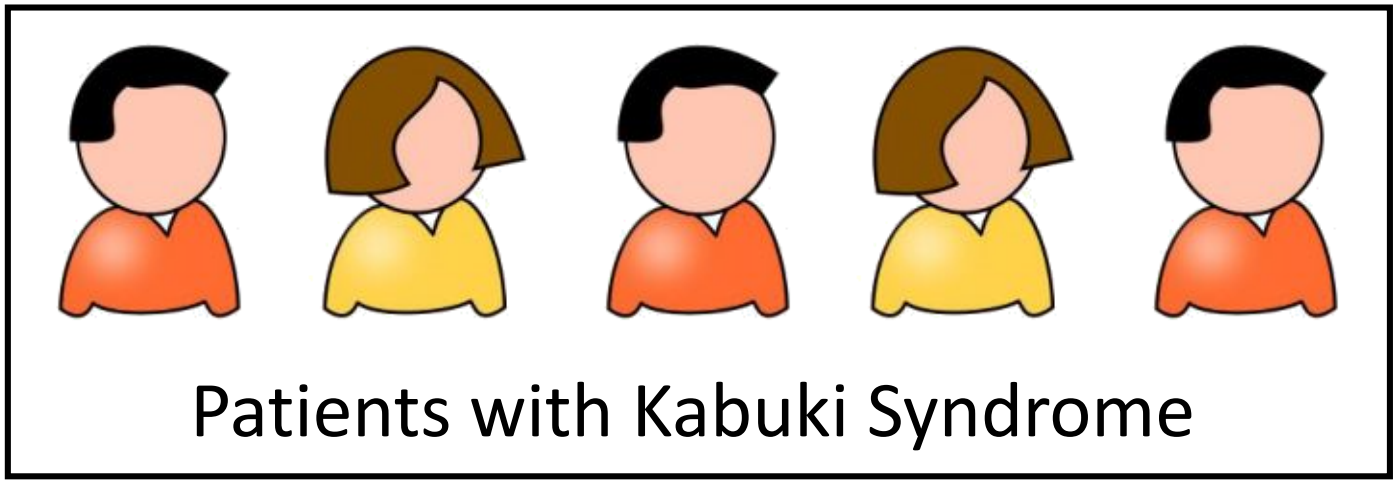
Known cause of disease

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh

MPC Protocol

Experiments on *real* patient data



Each patient has a list of 200-400 rare variants over ≈20,000 genes

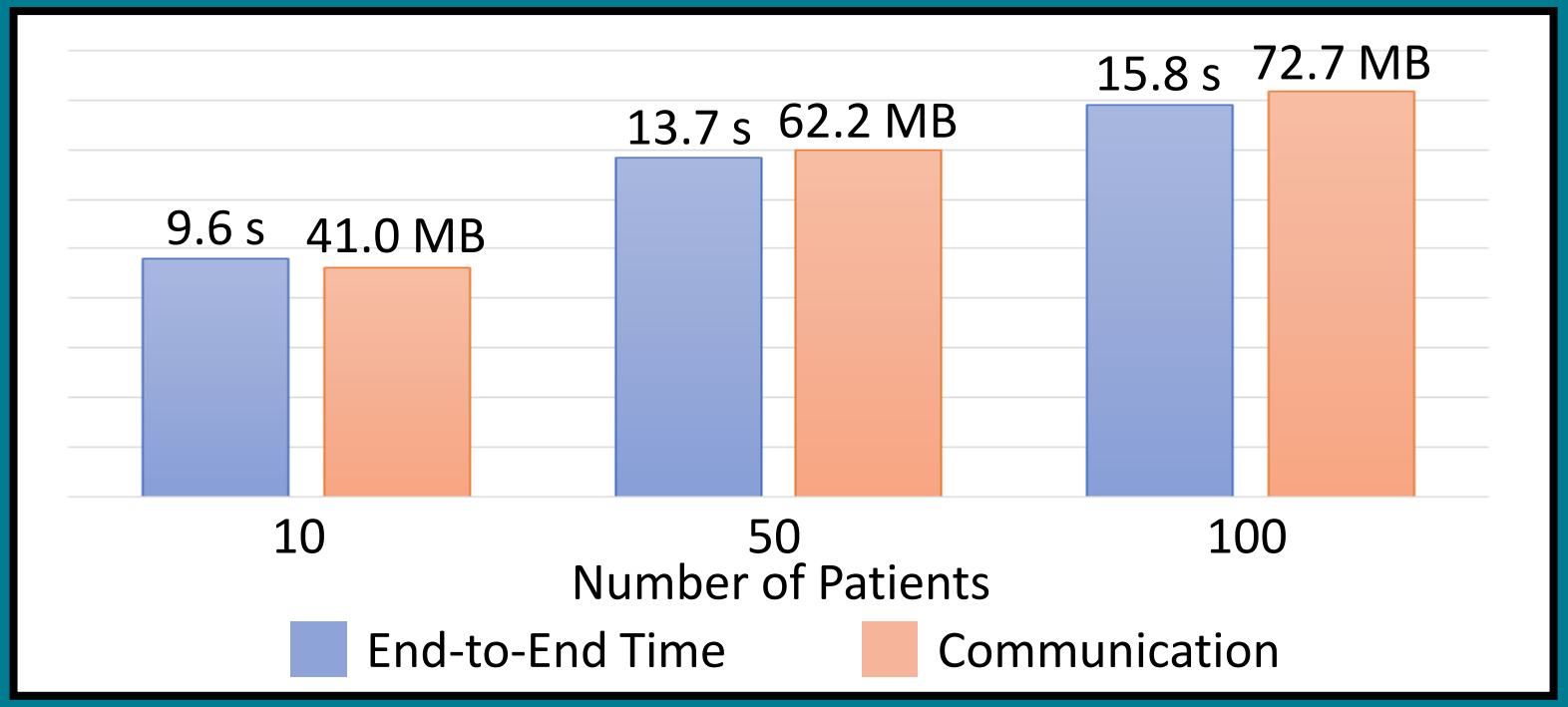
Top variants (sorted):
KMT2D, COL6A1, FLNB

Each hospital individually learns nothing about genomes

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh

Simulated two parties using two servers on Amazon EC2 (East Coast / West Coast):



Performance scales <u>logarithmically</u> with cohort size

Experiments on *real* patient data

Completes in under 10s

Top variants (sorted):
KMT2D, COL6A1, FLNB

Each hospital individually learns <u>nothing</u> about genomes

Private Genome Analysis [*Science* '17]

joint work with Boneh, Bejerano, Birgmeier, and Jagadeesh

MPC Protocol

Experiments on *real* patient data

Completes in under 10s

Top variants (sorted):
KMT2D, COL6A1, FLNB

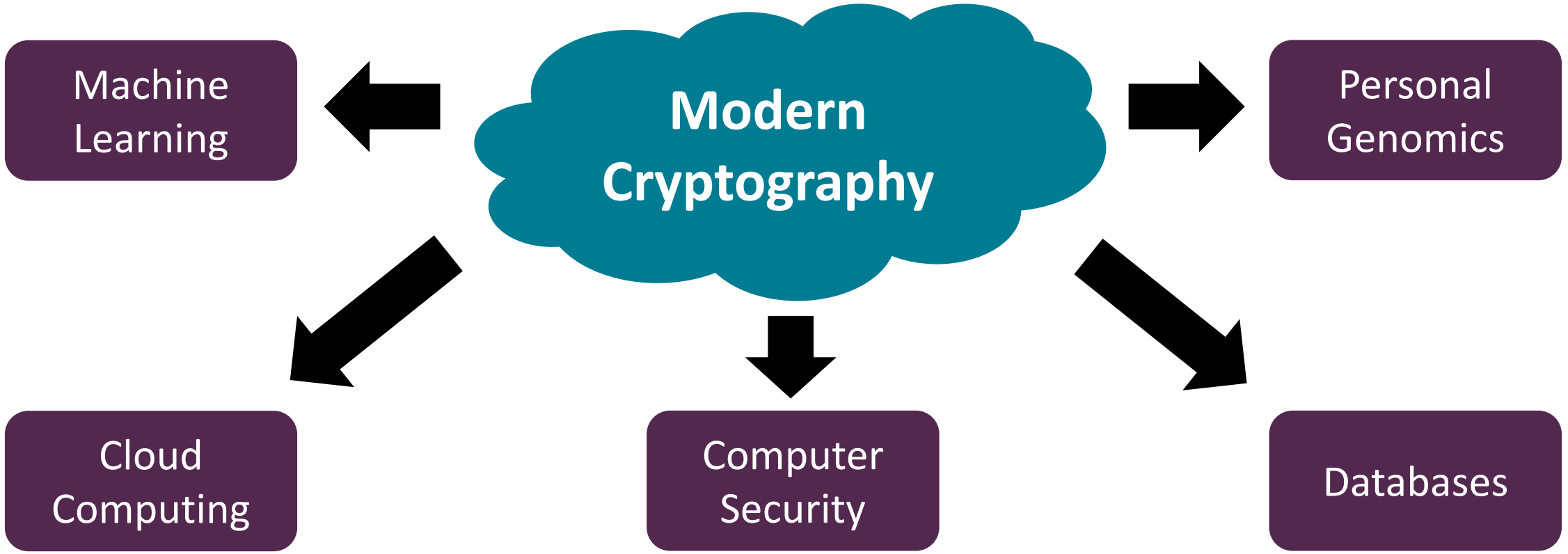
Area of growing interest: annual iDASH competition for developing solutions for privacy-preserving genomics

Upcoming work: privacy-preserving genome-wide association studies (GWAS) framework with tens of thousands of genomes [CWB18; *Nature Biotechnology*]

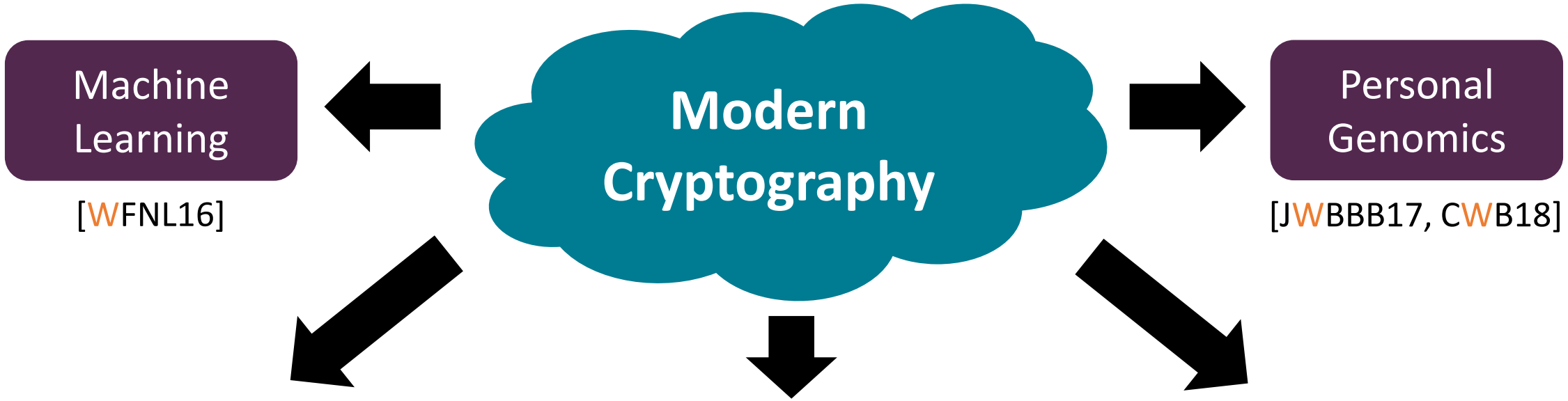
[Preliminary implementation won first place at iDASH 2015]

Each hospital individually learns nothing about genomes

Modern Cryptography

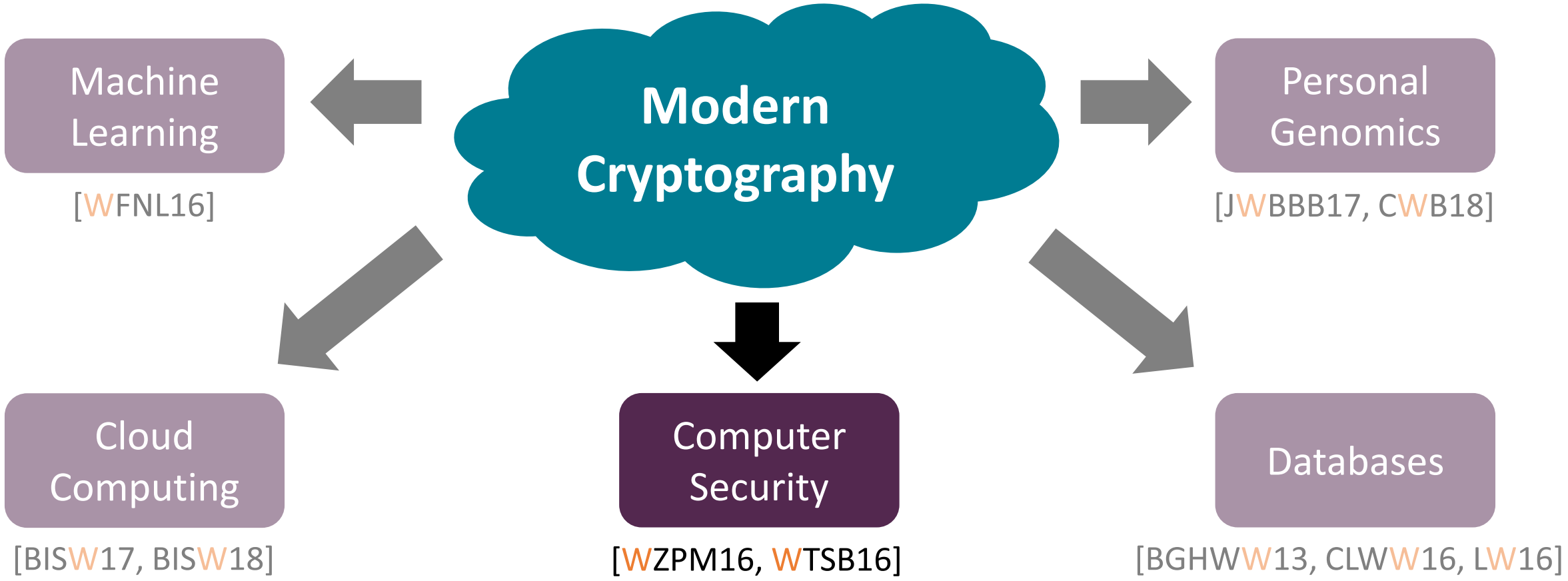


My Research from 10,000 Feet



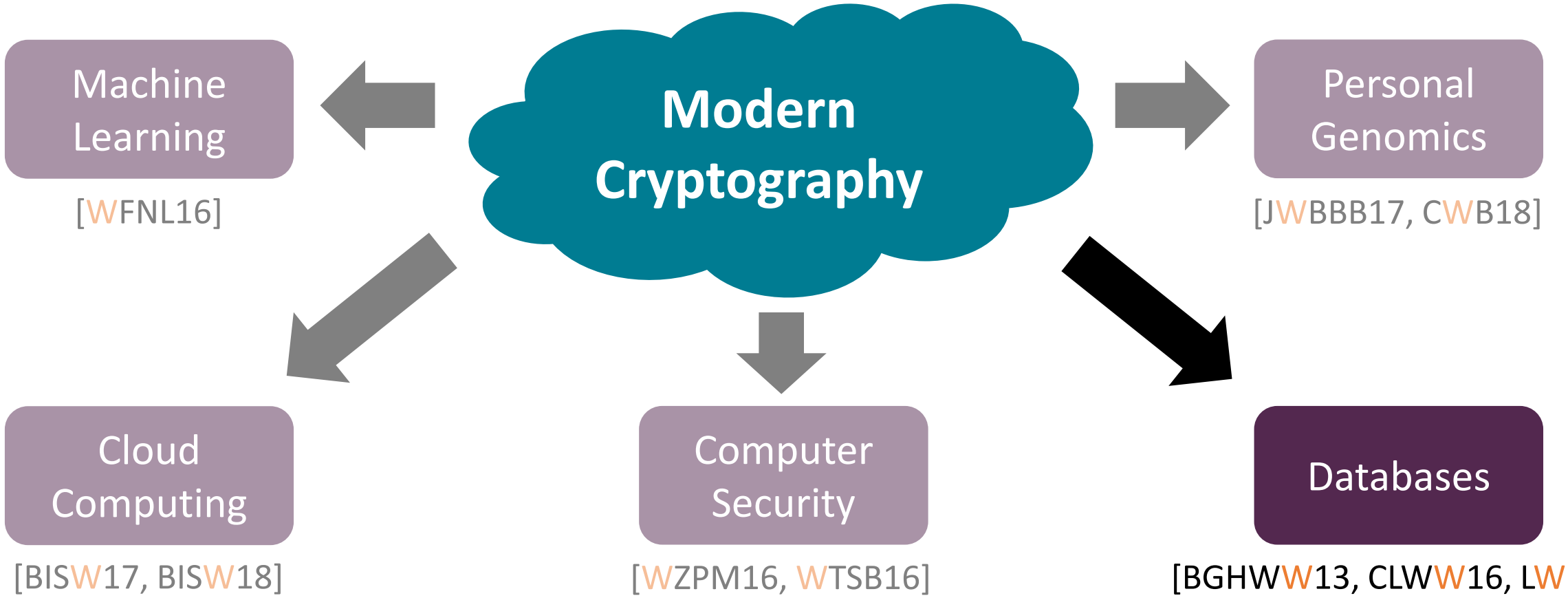
My Research from 10,000 Feet

How can we build more <u>secure</u> systems?



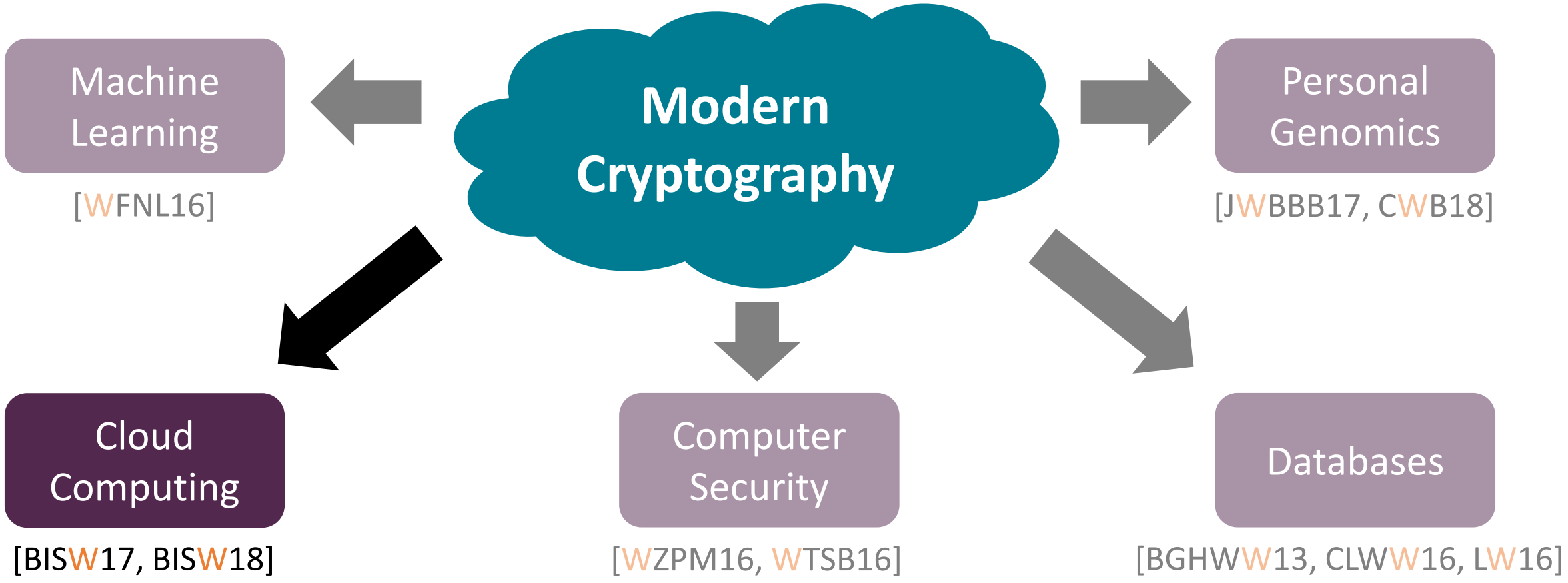
My Research from 10,000 Feet

How do we search on encrypted data?

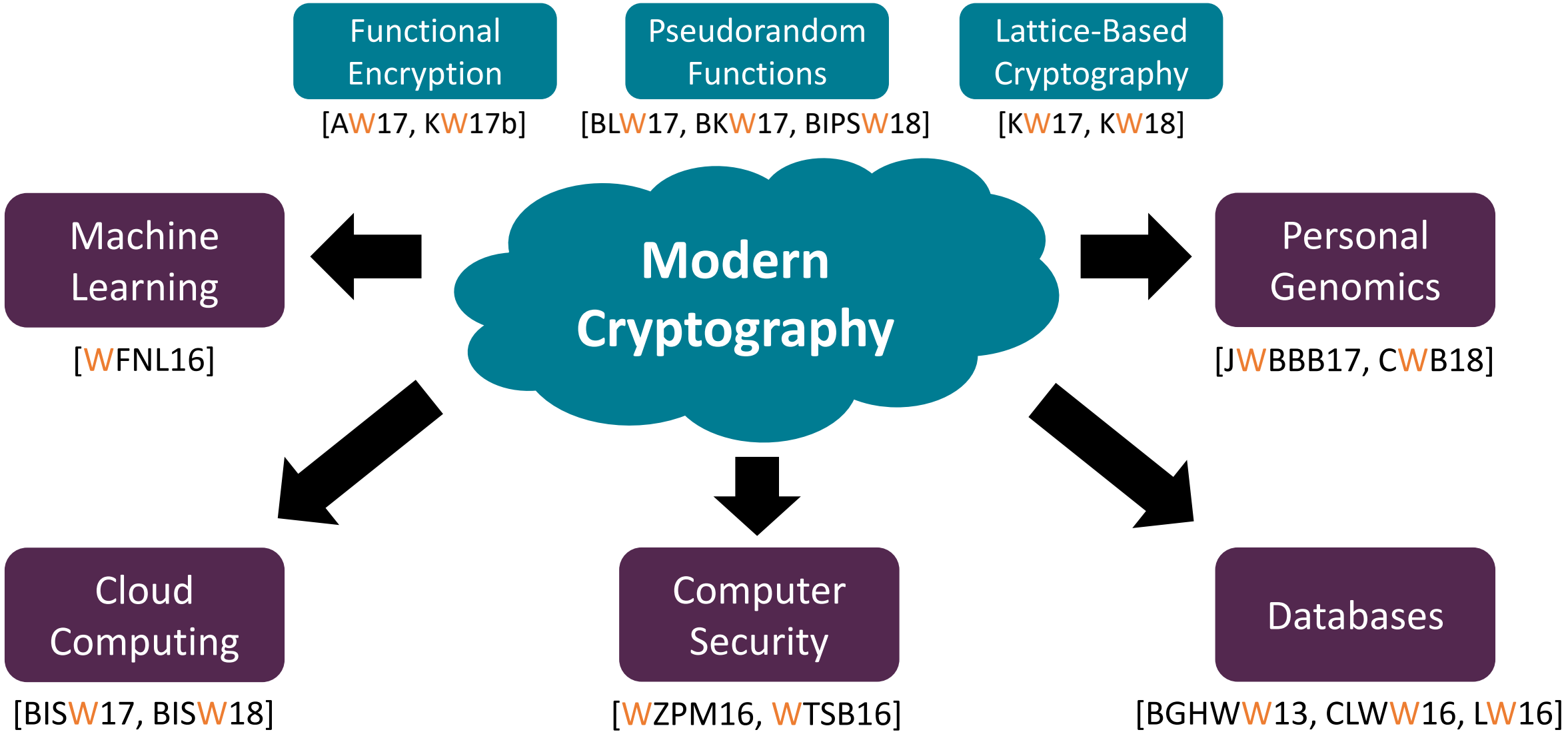


My Research from 10,000 Feet

How can a user <u>efficiently</u> verify the correctness of a complex computation?



My Research from 10,000 Feet



Talk Outline

Functional Encryption

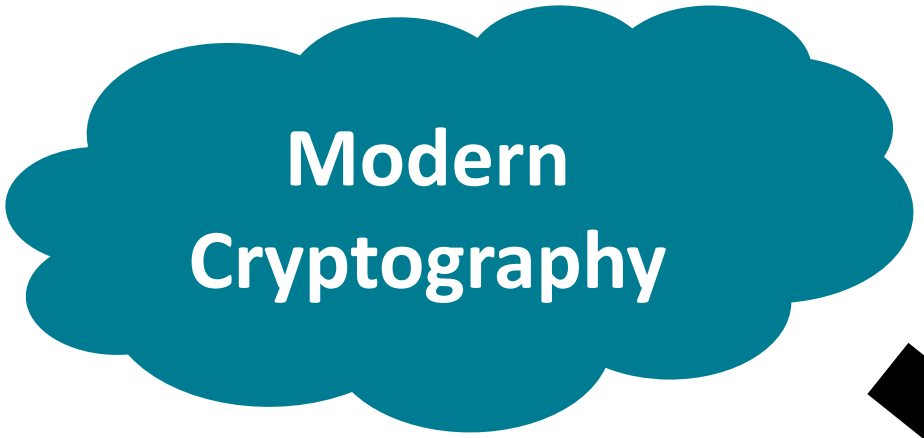
[AW17, KW17b]

Pseudorandom Functions

[BLW17, BKW17, BIPSW18]

Lattice-Based Cryptography

[KW17, KW18]



Personal Genomics

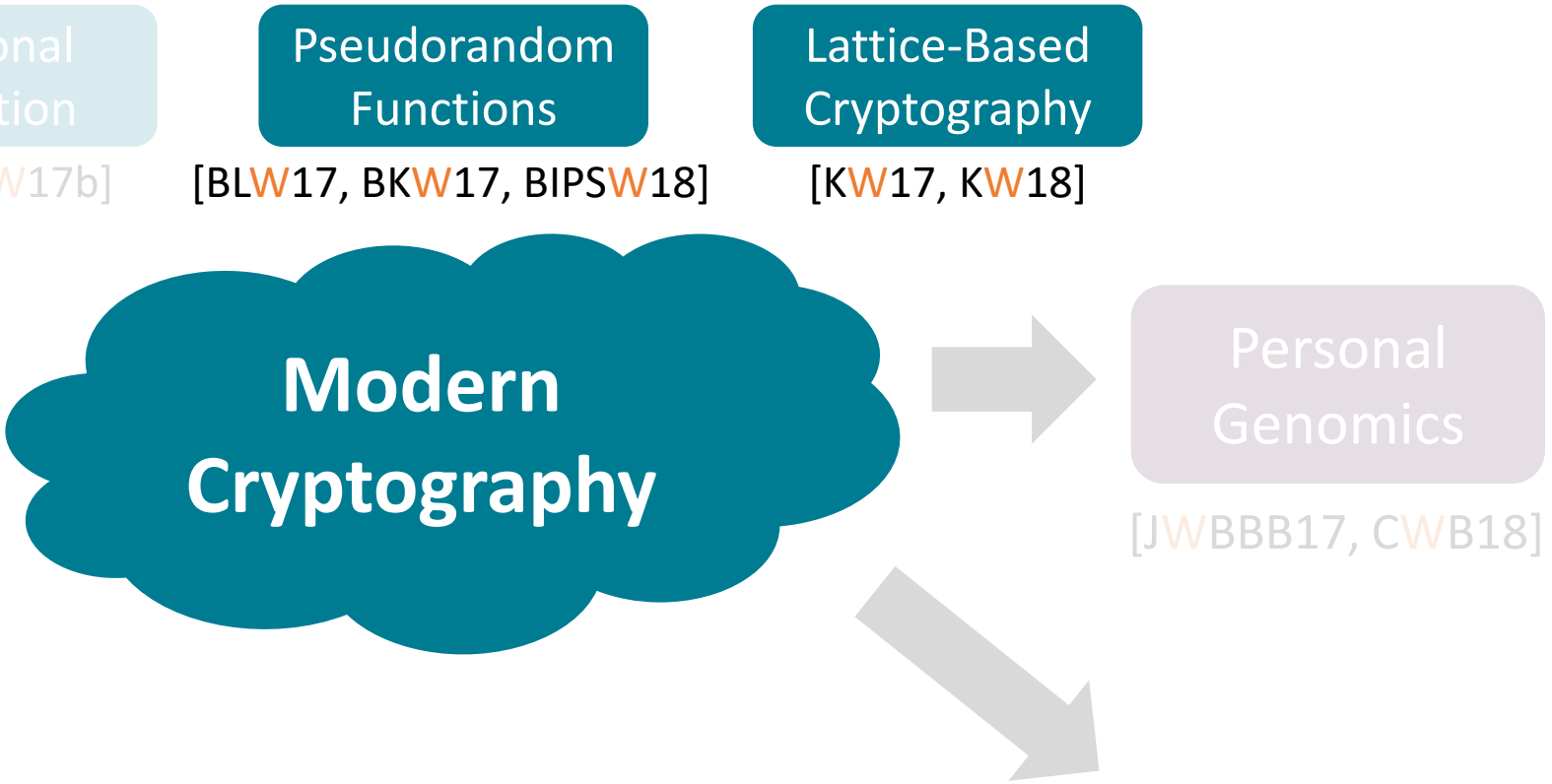
[JWBBB17, CWB18]

Part I: Searching on Encrypted Data

Databases

[BGHWW13, CLWW16, LW16]

Talk Outline

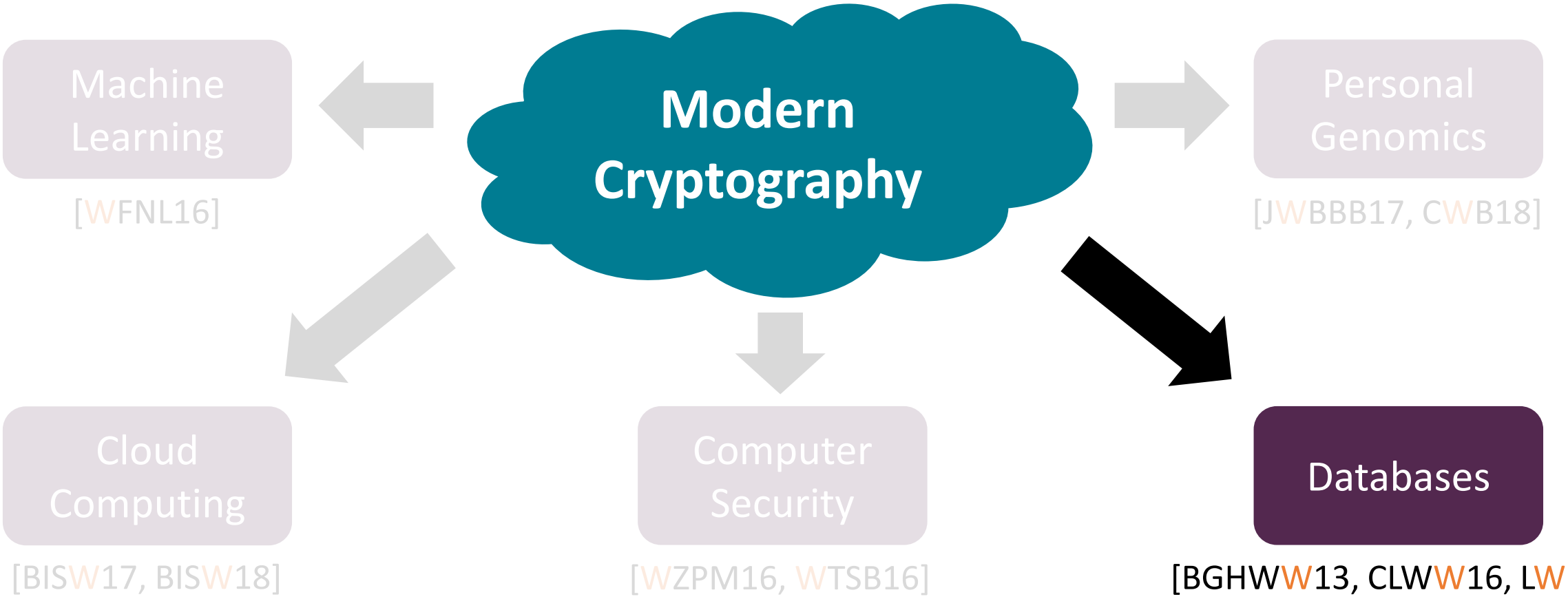


Part I: Searching on Encrypted Data

Part II: Software Watermarking

Part I: Searching on Encrypted Data

Main theme: Developing new cryptographic primitives that enable secure systems design



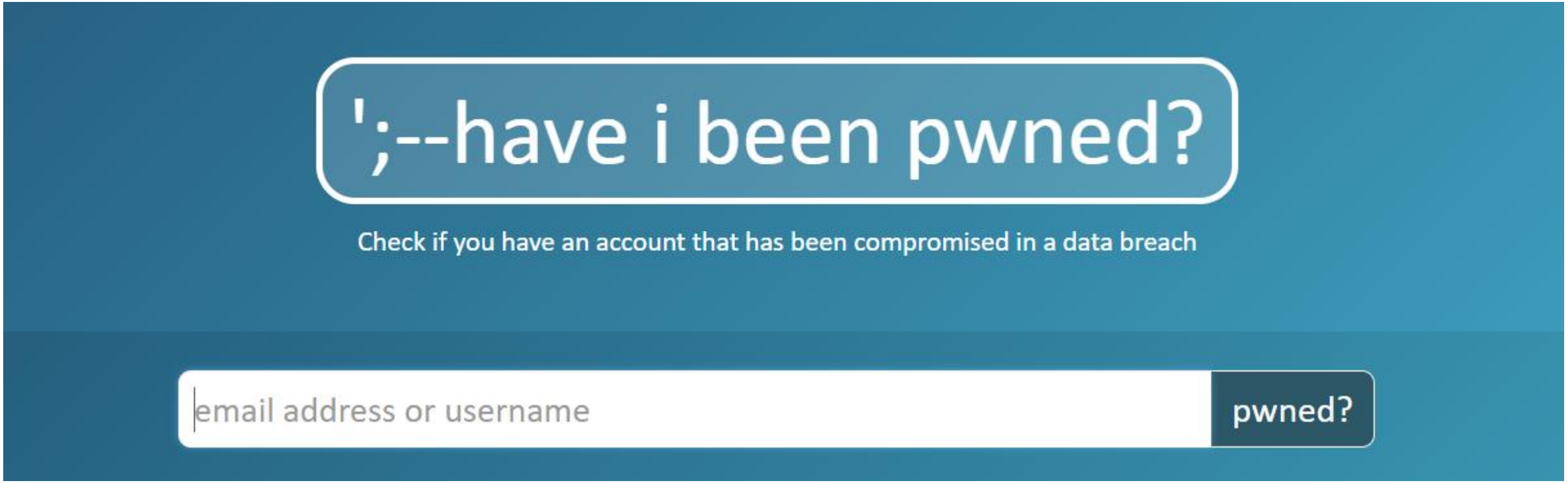
Searching on Encrypted Data

Entries	Database	Category	Dump Date
358,676,097	Myspace.com	Social Media	2013-06
153,004,874	Adobe.com	Software	2013-10
117,046,470	LinkedIn.com	Social Media	2012
77,039,888	Edmodo.com	Education	2017-05
68,743,269	Neopets.com	Gaming	2013-10
36,397,296	AshleyMadison.com	Dating	2015-07
16,500,334	Zomato.com	Food & Drink	2017-05
6,054,459	Xat.com	Chatroom	2015-11
5,960,654	Adobe.com Common Passwords	Software	2013-10

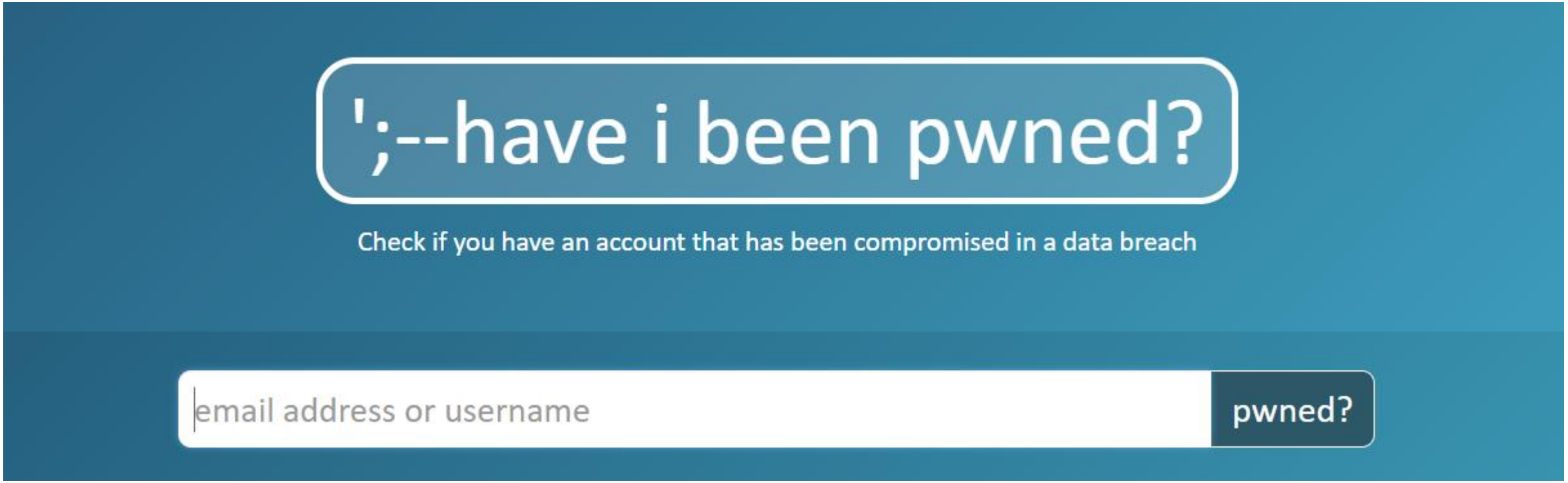
Database breaches have become the norm rather than the exception…

[Data taken from Vigilante.pw]

Searching on Encrypted Data



Why Not Encrypt?



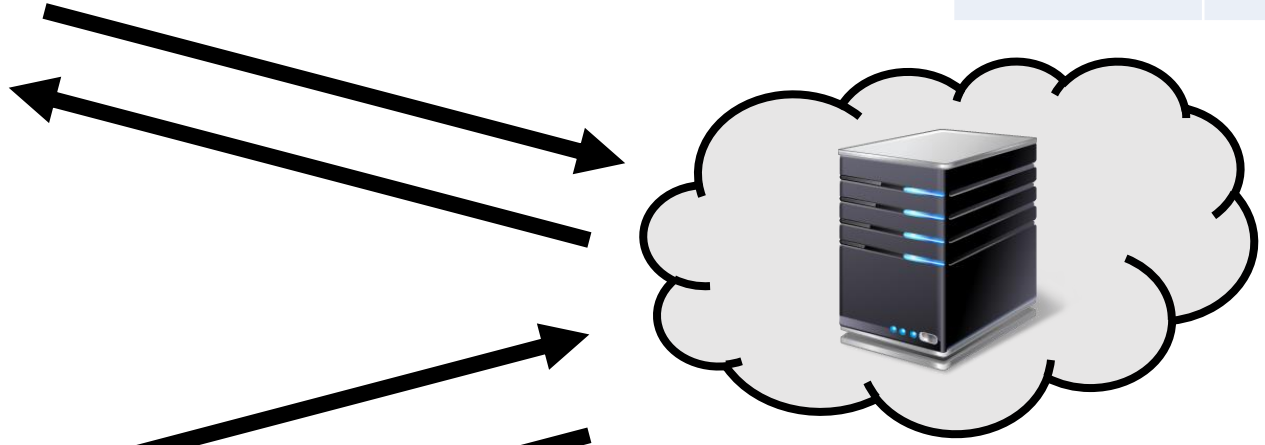
"Because it would have hurt Yahoo's ability to index and search messages to provide new user services"

– Jeff Bonforte (Yahoo SVP)

Searching on Encrypted Data

Any client (e.g., web client, employee) who hold a secret key can query the database

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304



encrypted database

database server
(hosted in the cloud)

Security Against "Snapshot Adversaries"

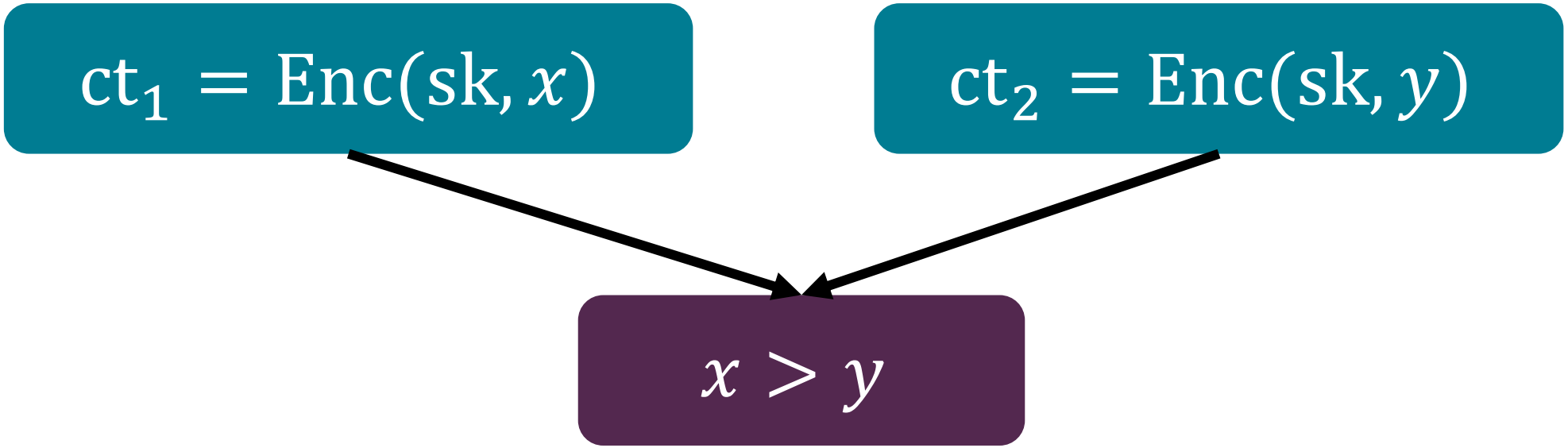
ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

Adversary breaks into the database server and steals the contents of the database on disk (i.e., obtains a "snapshot" of the database)

database server (hosted in the cloud)

Order-Revealing Encryption [BCLO09, BLRSZZ15]

secret-key encryption scheme

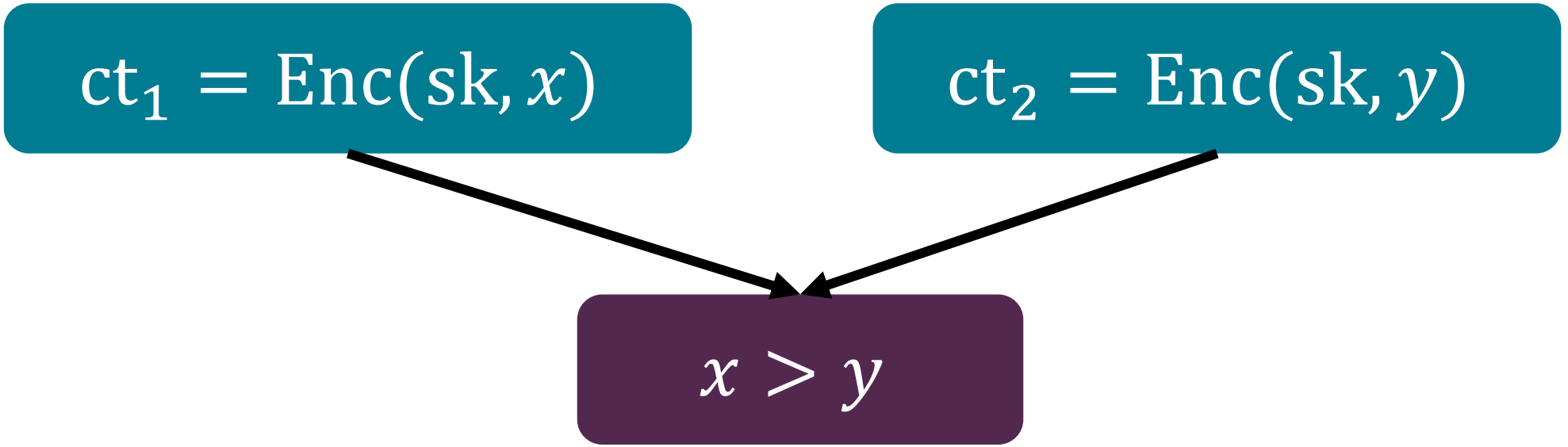


public comparison function for ciphertexts

Best-possible security: ciphertexts hide everything other than the ordering of the values

Order-Revealing Encryption [BCLO09, BLRSZZ15]

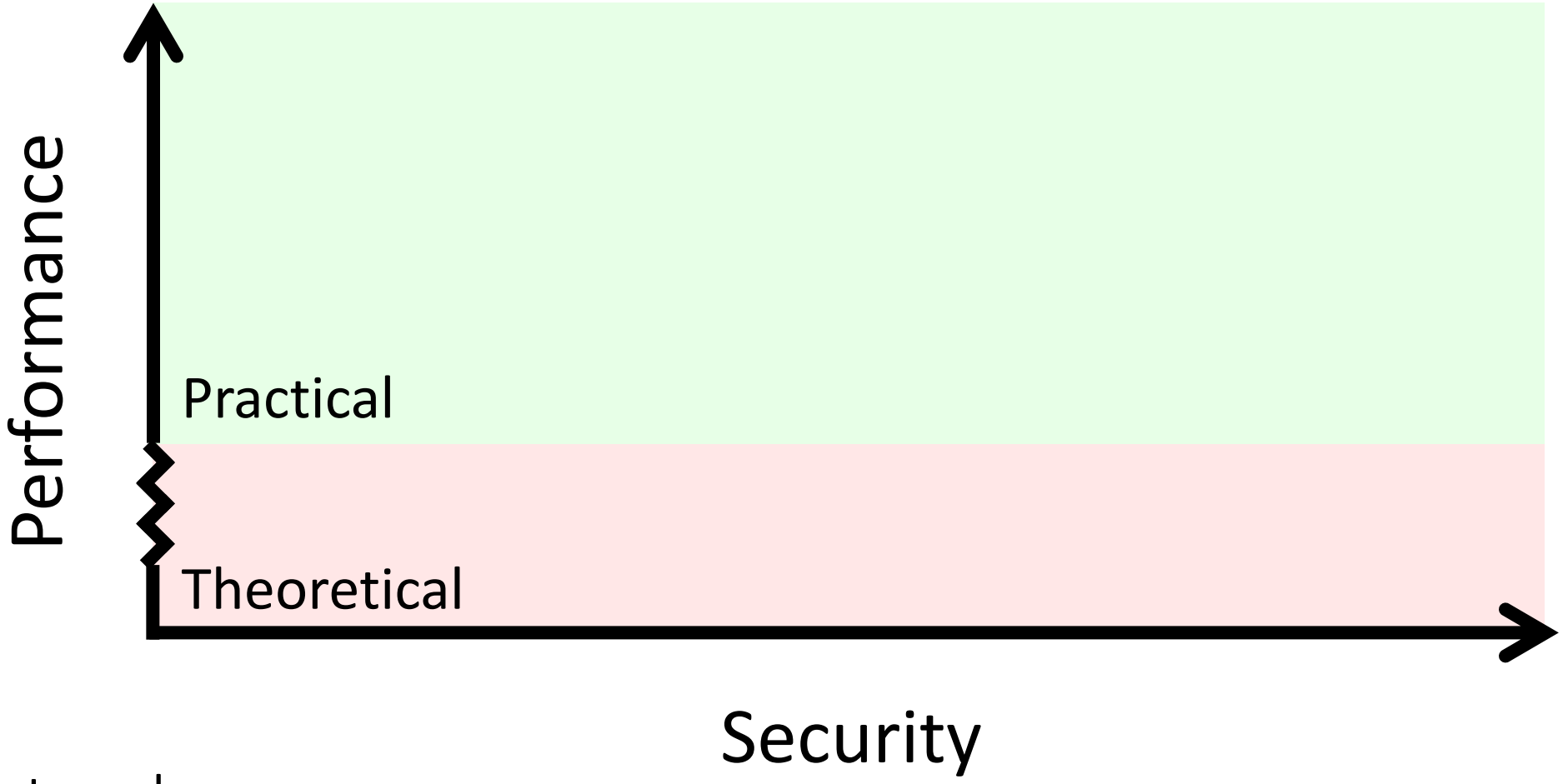
secret-key encryption scheme



public comparison function for ciphertexts

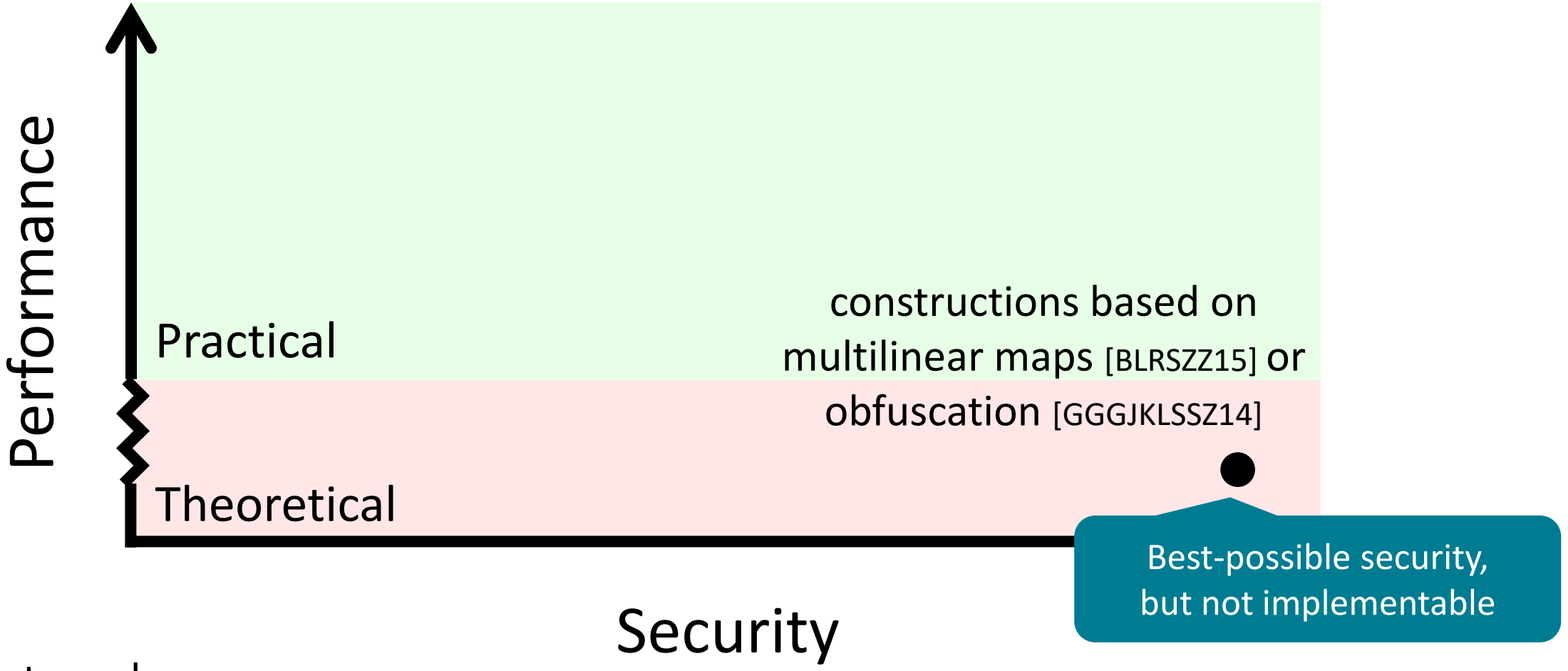
Enables queries on encrypted data without making significant changes to existing database architectures

Existing Approaches



Not drawn to scale

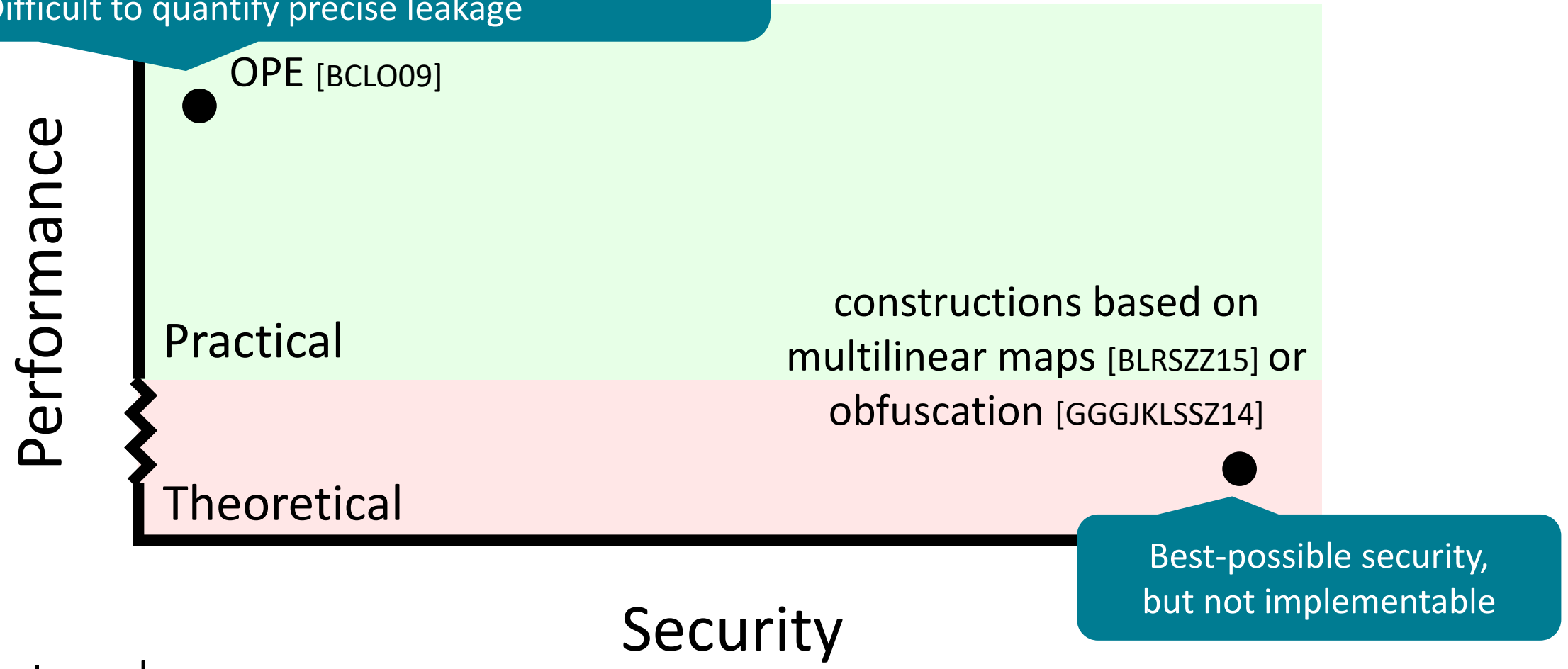
Existing Approaches



Not drawn to scale

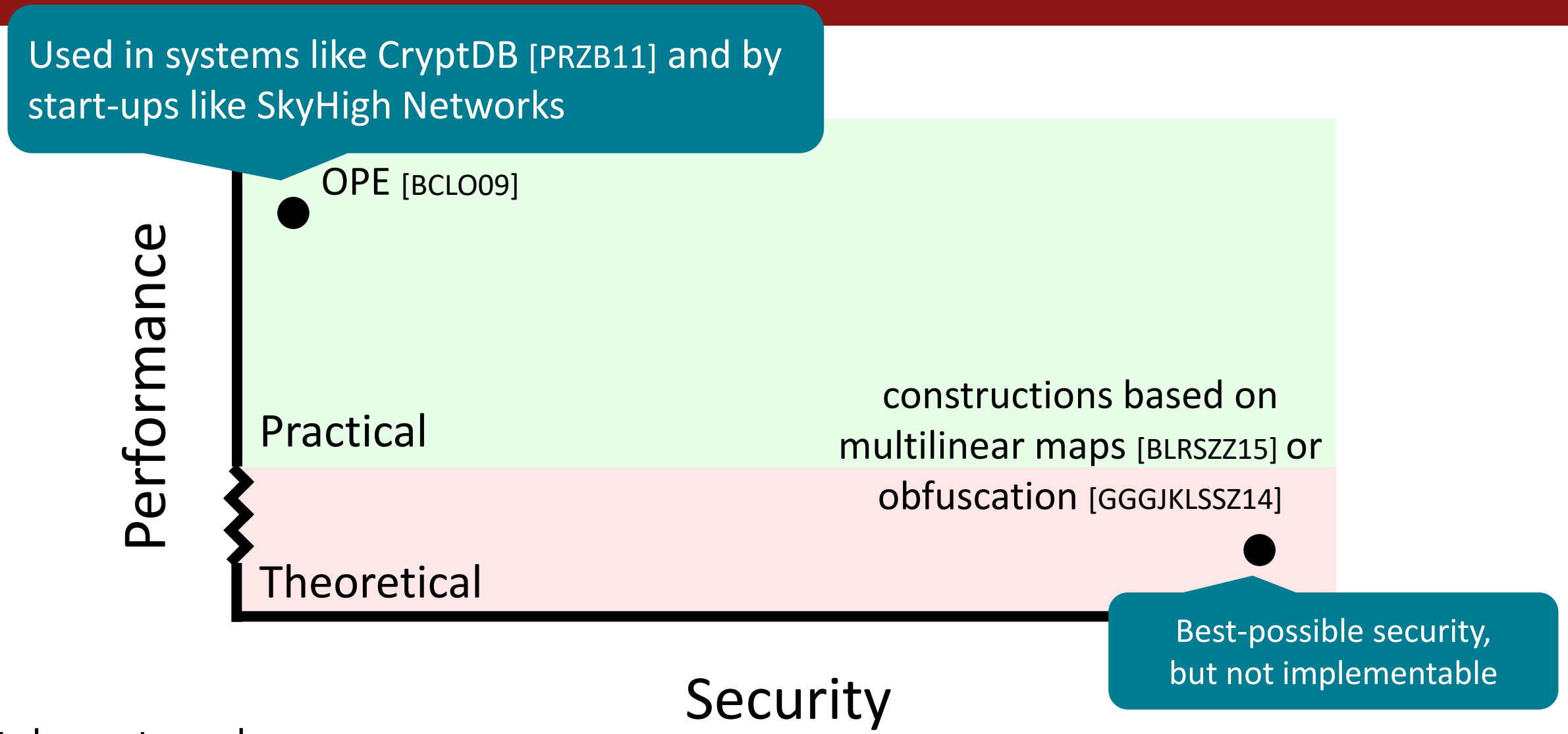
Existing Approaches

Very efficient, but has additional leakage:
- Ciphertexts reveal *half* of the bits of the plaintext
- Difficult to quantify precise leakage



Not drawn to scale

Existing Approaches

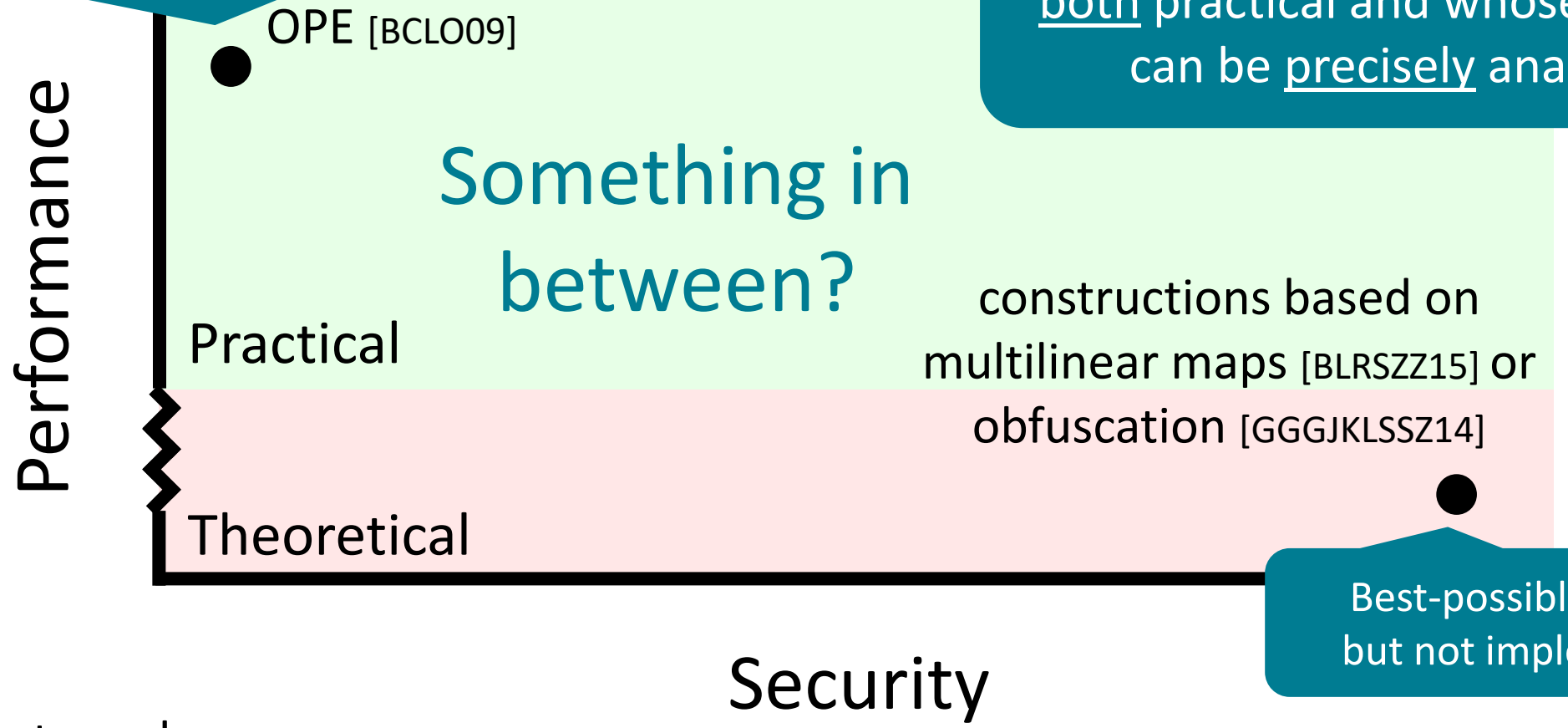


Not drawn to scale

Existing Approaches

Used in systems like CryptDB [PRZB11] and by start-ups like SkyHigh Networks

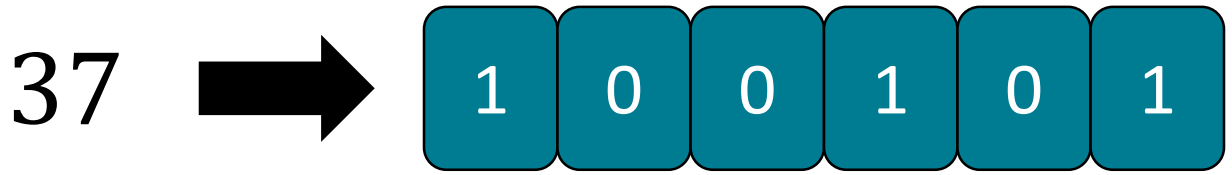
Goal: New notion of ORE that is both practical and whose security can be precisely analyzed



Not drawn to scale

A Simple ORE Construction [*FSE* '16]

joint work with Chenette, Lewi, and Weis



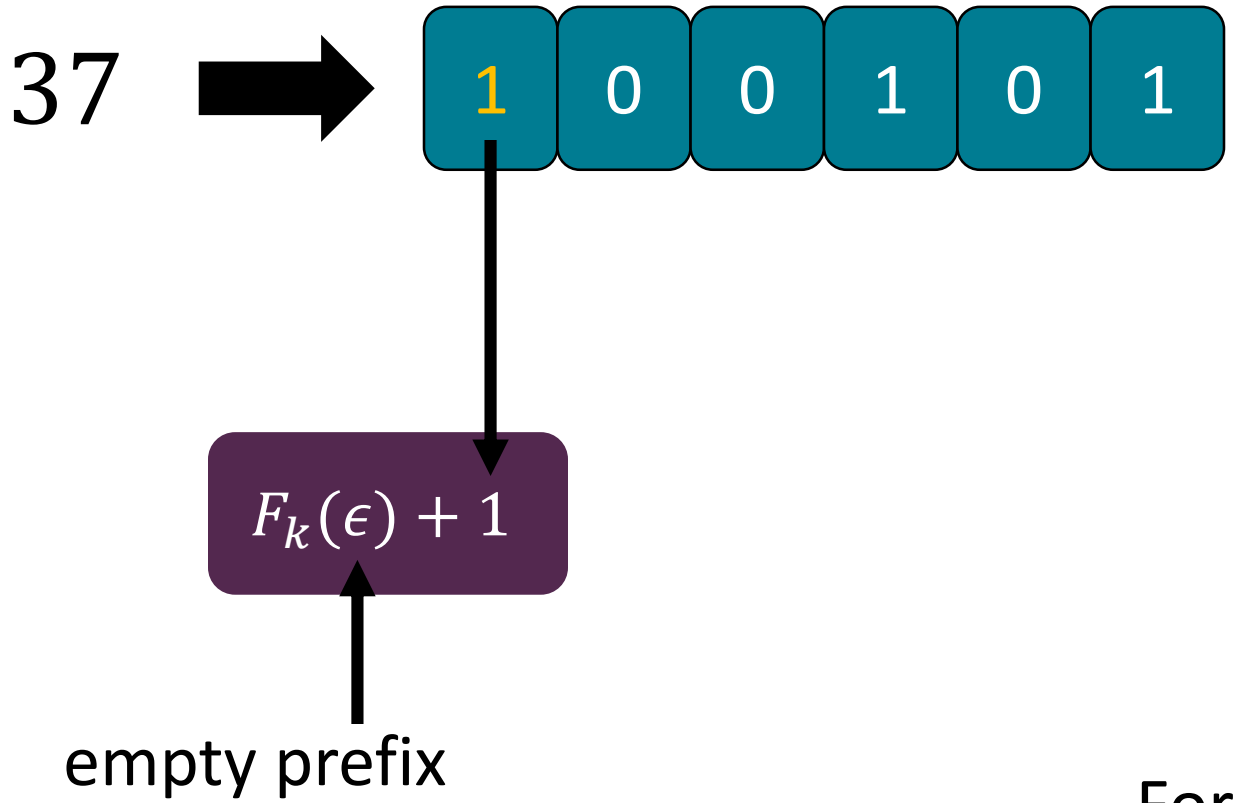
Pseudorandom function (PRF): function whose input-output behavior looks like that of a random function

For each index i, apply a PRF (e.g., AES) to the first $i-1$ bits, then add $b_i \pmod 3$

F_k: $\{0,1\}^* \rightarrow \{0,1,2\}$

A Simple ORE Construction [*FSE* '16]

joint work with Chenette, Lewi, and Weis

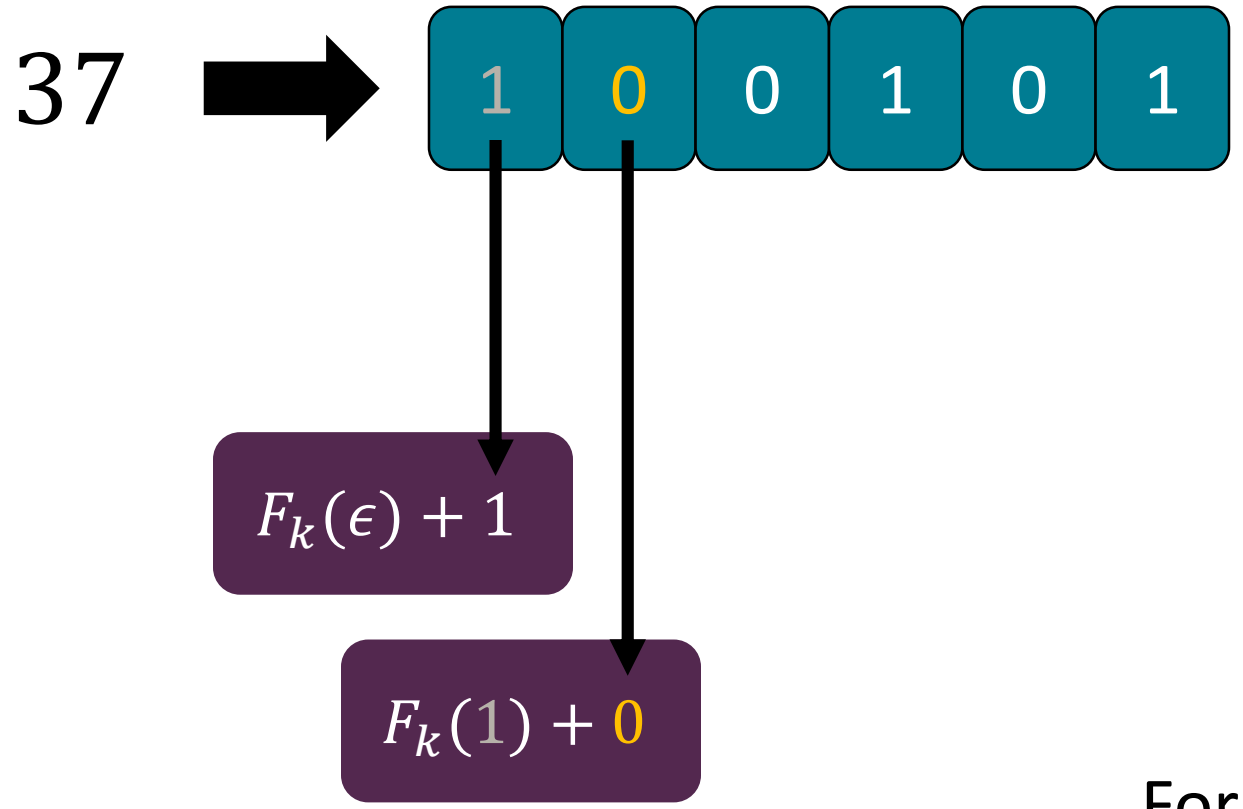


F_k: $\{0,1\}^* \rightarrow \{0,1,2\}$

For each index i, apply a PRF (e.g., AES) to the first $i-1$ bits, then add $b_i \pmod 3$

A Simple ORE Construction [*FSE* '16]

joint work with Chenette, Lewi, and Weis

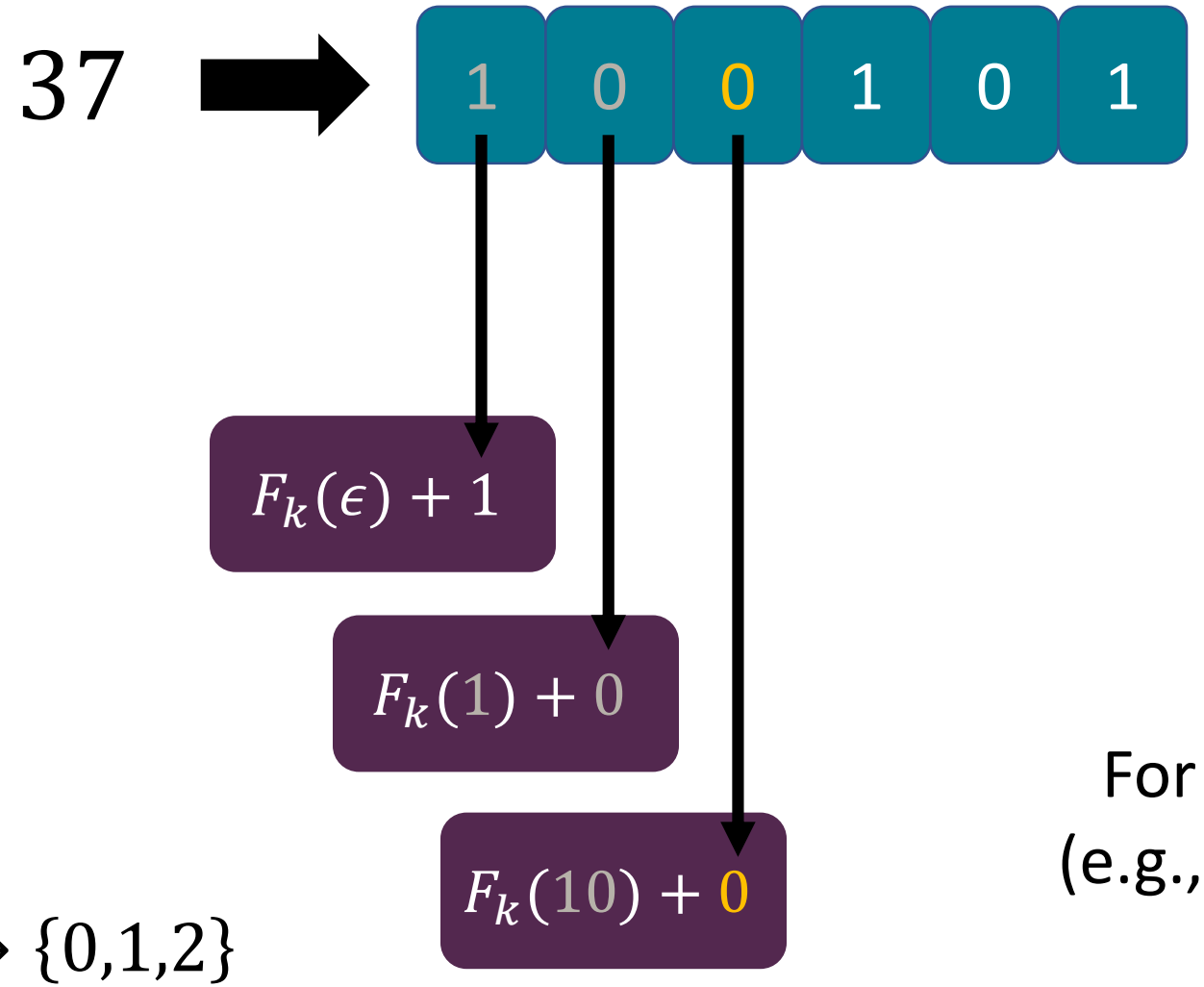


F_k: $\{0,1\}^* \rightarrow \{0,1,2\}$

For each index i, apply a PRF (e.g., AES) to the first $i-1$ bits, then add $b_i \pmod 3$

A Simple ORE Construction [*FSE* '16]

joint work with Chenette, Lewi, and Weis

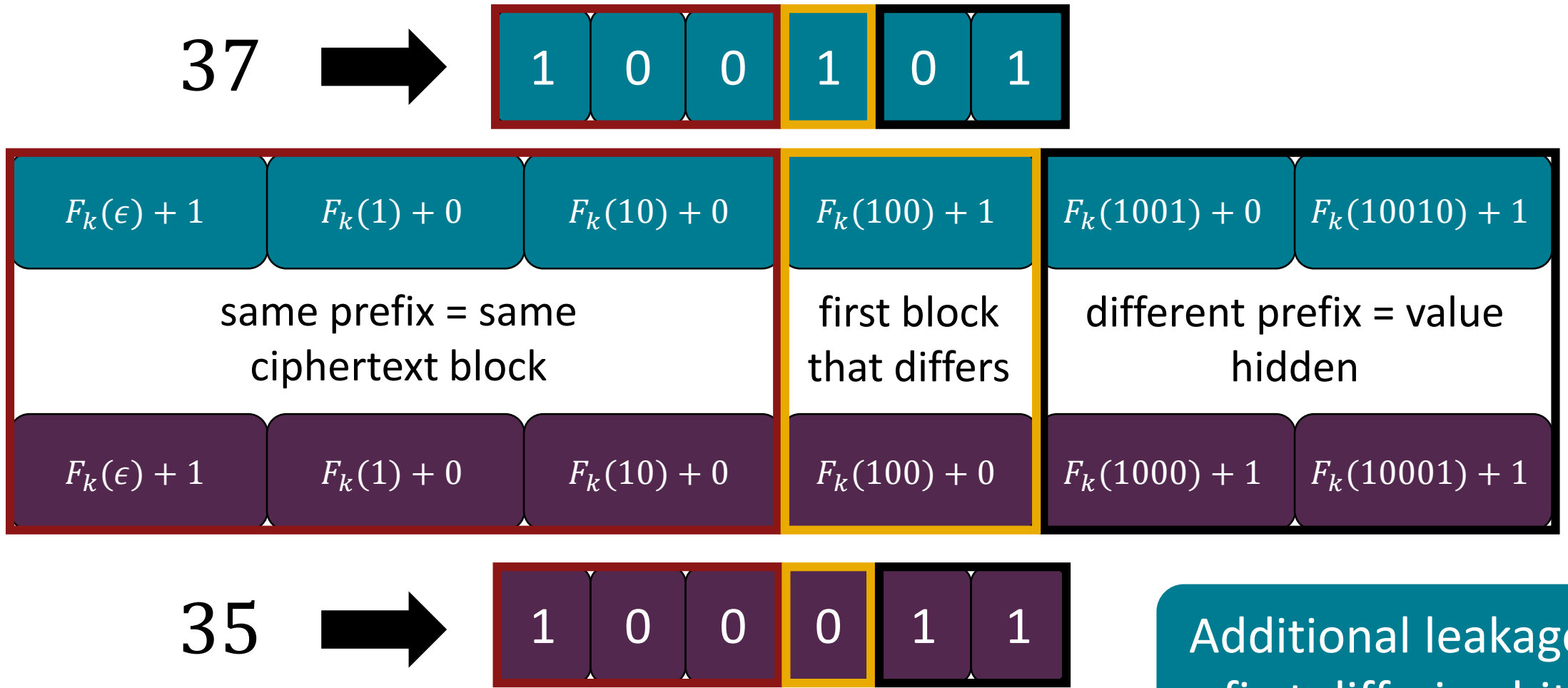


F_k: $\{0,1\}^* \rightarrow \{0,1,2\}$

For each index i, apply a PRF (e.g., AES) to the first $i-1$ bits, then add $b_i \pmod 3$

A Simple ORE Construction [*FSE* '16]

joint work with Chenette, Lewi, and Weis

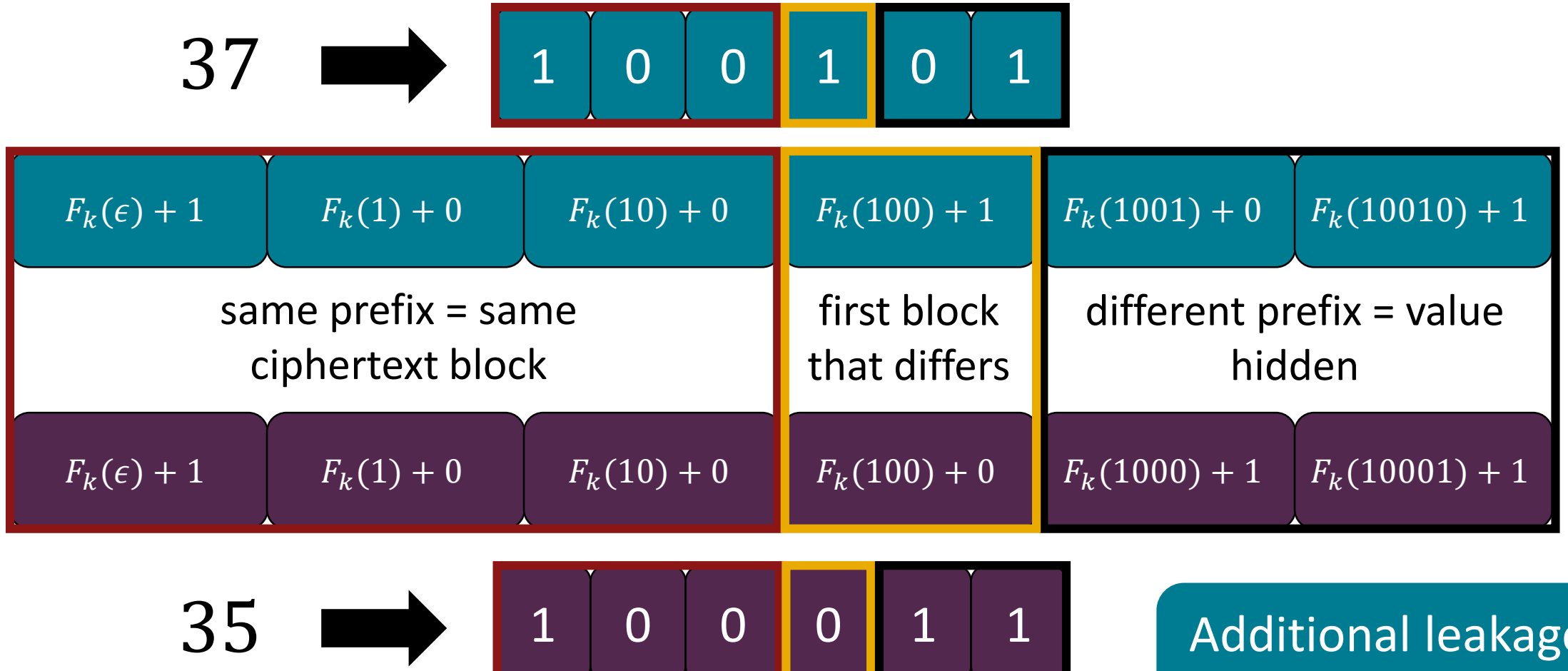


Additional leakage: first differing bit

Recall: all additions happen modulo 3

A Simple ORE Construction [*FSE* '16]

joint work with Chenette, Lewi, and Weis



Additional leakage: first differing bit

Key insight: Embed comparisons into $\mathbb{Z}_3$

Inference Attacks [NKW15, DDC16, GSBNR17]

ID	Name	Age	Zip Code
wpjOos	2wzXW8	SqX9l9	KqLUXE
XdXdg8	y9GFpS	gwilE3	MJ23b7
P6vKhW	EgN0Jn	S0pRJe	aTaeJk
orJRe6	KQWy9U	tPWF3M	4FBEO0

encrypted database

+

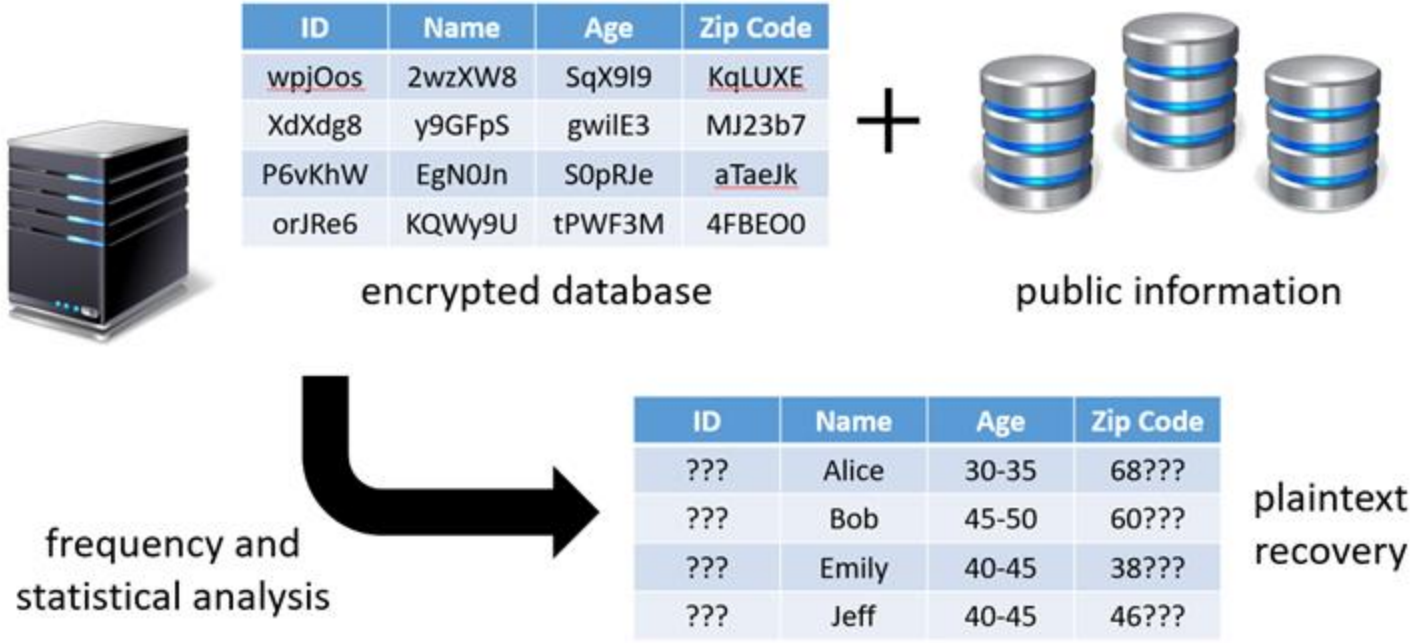
public information

frequency and statistical analysis

ID	Name	Age	Zip Code
???	Alice	30-35	68???
???	Bob	45-50	60???
???	Emily	40-45	38???
???	Jeff	40-45	46???

plaintext recovery

Inference Attacks [NKW15, DDC16, GSBNR17]



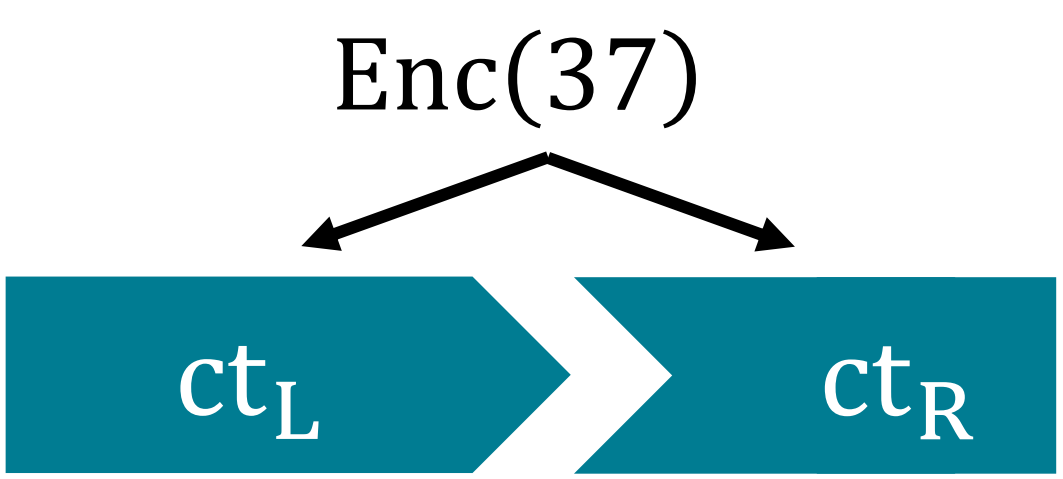
ORE schemes reveal order of ciphertexts and thus, are vulnerable to offline inference attacks

Can we extend ORE to defend against offline inference attacks?

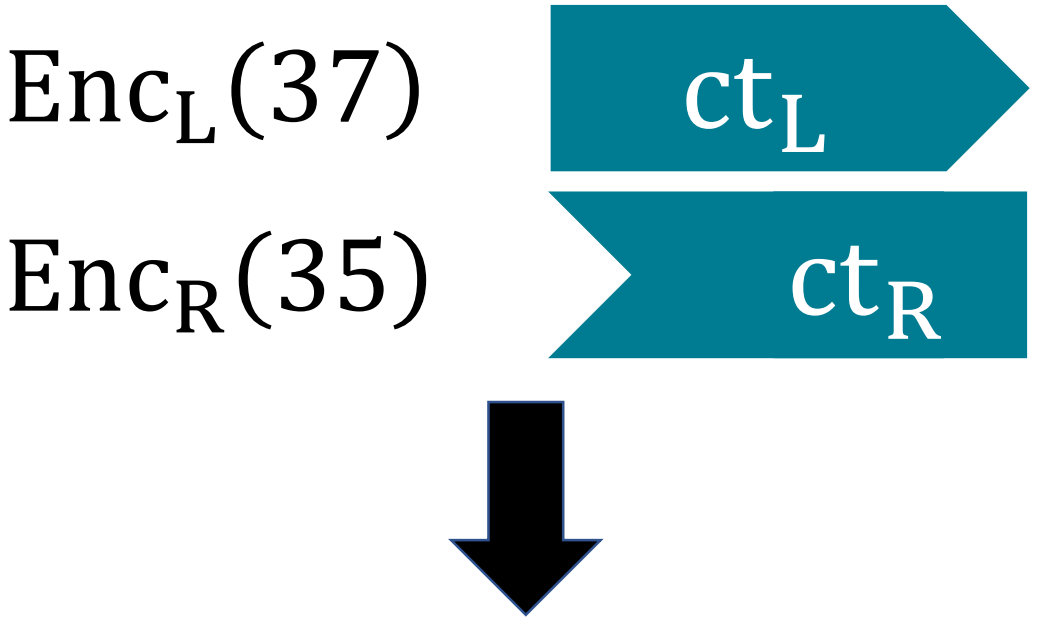
Defending Against Inference Attacks [*CCS* '16]

joint work with Lewi

Key primitive: order-revealing encryption scheme where ciphertexts have a decomposable structure



ciphertexts naturally split into two components



greater than

Defending Against Inference Attacks [*CCS* '16]

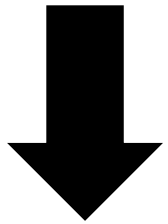
joint work with Lewi

$\mathrm{Enc}_L(37)$ → ct_L

$\mathrm{Enc}_R(35)$ → ct_R

comparison can be performed between left ciphertext and right ciphertext

right ciphertexts reveal <u>nothing</u> about underlying messages!



robustness against offline inference attacks!

Encrypted Range Queries [*CCS* '16]

joint work with Lewi

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

build encrypted index

Age	ID
$\mathrm{Enc}_R(31)$	Enc(0)
$\mathrm{Enc}_R(41)$	Enc(2)
$\mathrm{Enc}_R(45)$	Enc(3)
$\mathrm{Enc}_R(47)$	Enc(1)

store right ciphertexts in sorted order

record IDs encrypted under independent key

Name	ID
Enc(Alice)	Enc(0)

Age	ID
Enc(31)	Enc(0)

Zip Code	ID
$\mathrm{Enc}_R(38655)$	Enc(2)
$\mathrm{Enc}_R(46304)$	Enc(3)
$\mathrm{Enc}_R(60015)$	Enc(1)
$\mathrm{Enc}_R(68107)$	Enc(0)

separate index for each searchable column, and using different ORE keys

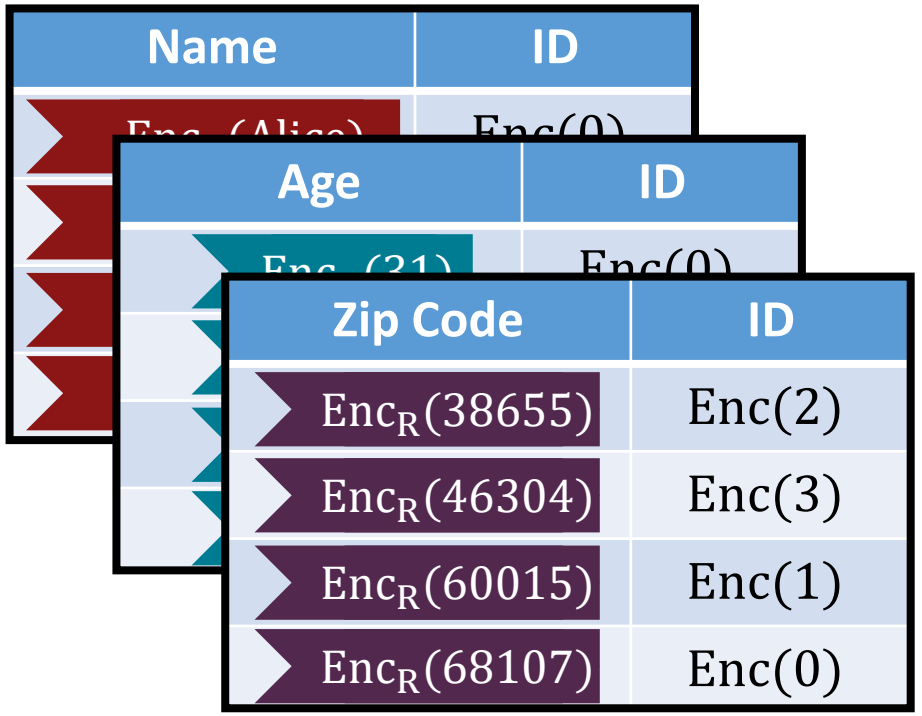
Encrypted Range Queries [*CCS* '16]

joint work with Lewi

Encrypted database:

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

columns (other than ID) are encrypted using standard encryption scheme



encrypted search indices

to perform range query, client provides left ciphertexts corresponding to its range

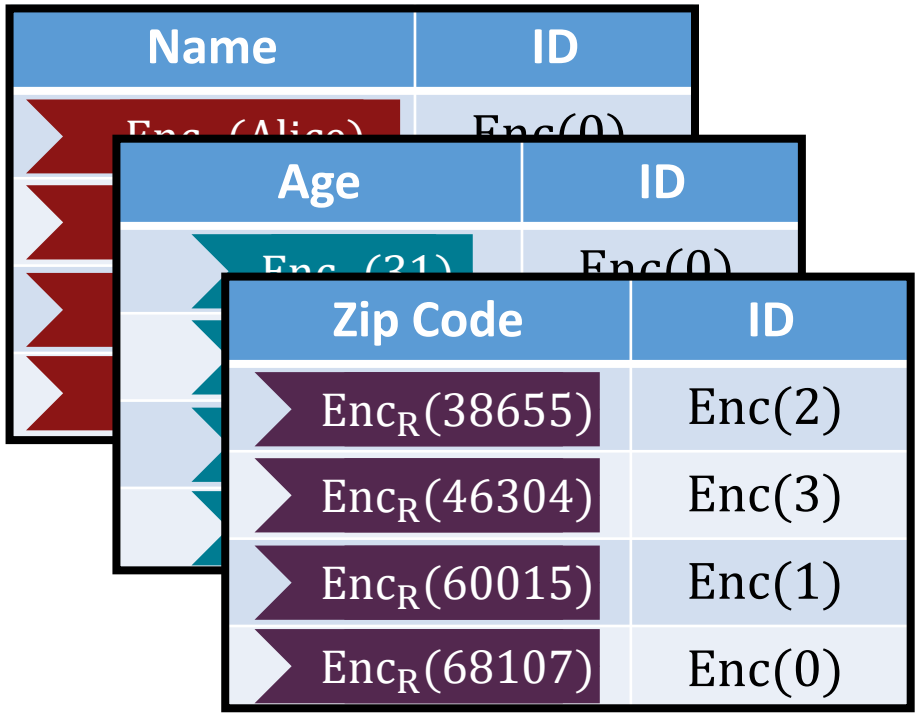
Encrypted Range Queries [*CCS* '16]

joint work with Lewi

Encrypted database:

ID	Name	Age	Zip Code
0	Alice	31	68107
1	Bob	47	60015
2	Emily	41	38655
3	Jeff	45	46304

Encrypted database hides the contents!



encrypted search indices

Performance Comparison

Scheme	Encrypt (μs)	Compare (μs)	$\|ct\|$ (bytes)	Security
[BCLO09] OPE	$> 10^3$	0.36	8	Leaks half of the bits
[CLWW16] ORE	2.06	0.48	8	Leaks first-differing bit
[LW16] ORE	54.87	0.63	224	Left-right security
5Gen ORE [LMAC+16]	$> 10^9$	$> 10^8$	$> 10^9$	Best-possible security (80 bits of security)

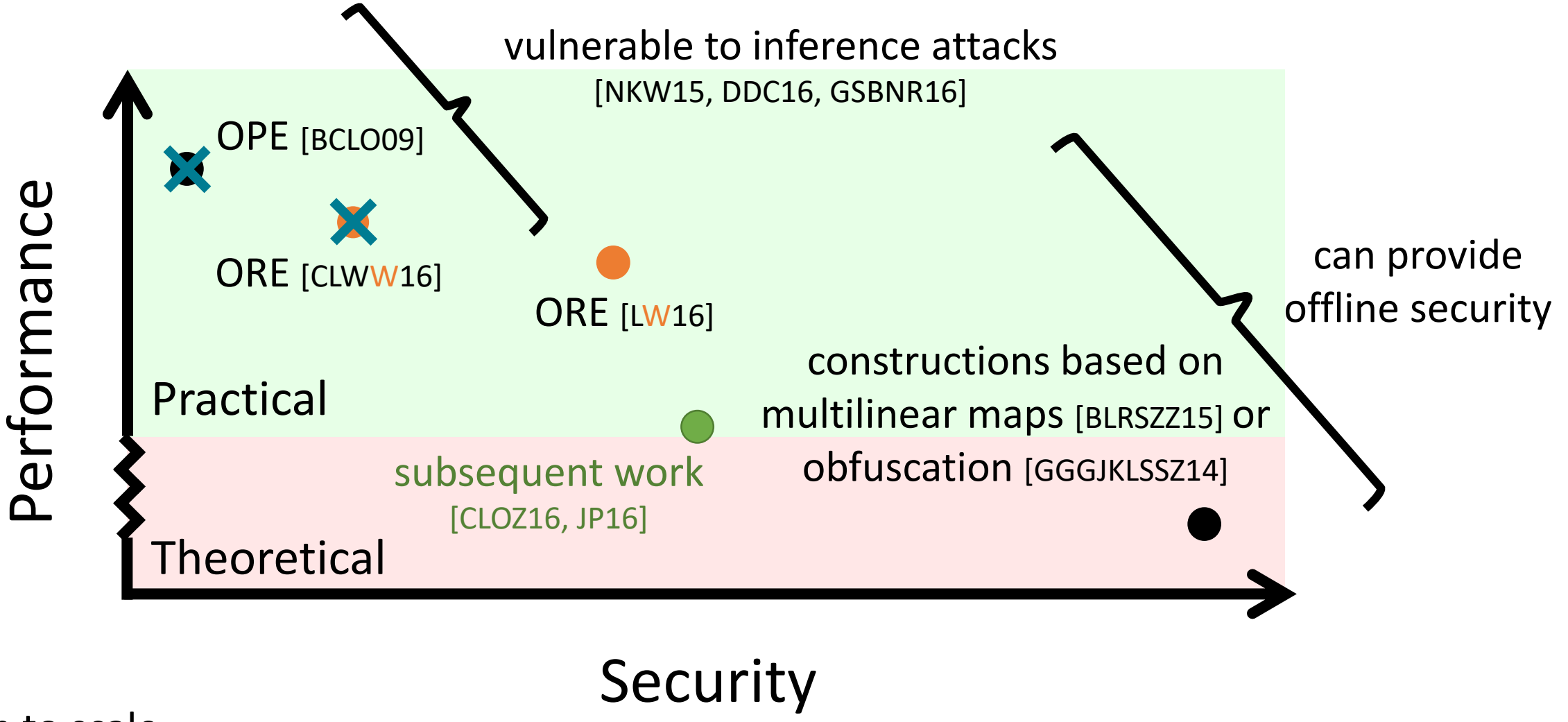
Measurements for encrypting 32-bit integers (with 128 bits of security)

Performance Comparison

Scheme	Encrypt (μs)	Compare (μs)	$\|ct\|$ (bytes)	Security
[BCLO09] OPE	$> 10^3$	0.36	8	Leaks half of the bits
[CLWW16] ORE	2.06	0.48	8	Leaks first-differing bit
[LW16] ORE	54.87	0.63	224	Left-right security
5Gen ORE [LMAC+16]	$> 10^9$	$> 10^8$	$> 10^9$	Best-possible security (80 bits of security)

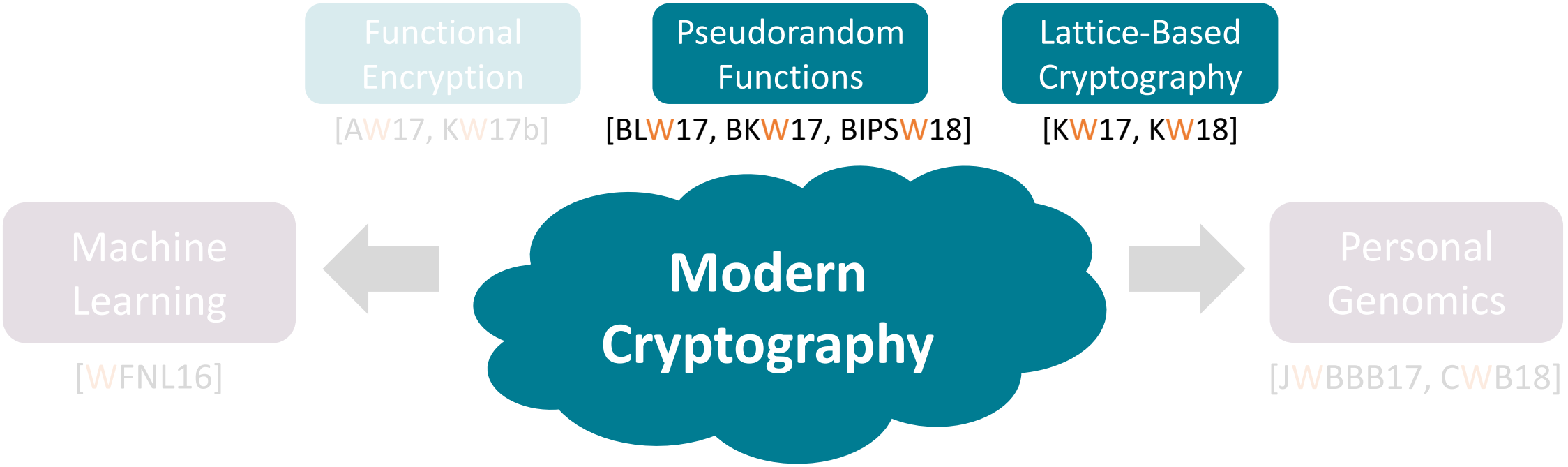
The [LW16] scheme is 65x faster than OPE, but ciphertexts are 30x longer. Security is substantially better.

The Landscape of ORE



Not drawn to scale

Part II: Watermarking Software



Main theme: Realizing complex cryptographic functionalities from simple assumptions

Watermarking Software

How do we prove ownership of software?

PolicyNodeImpl.java (Java version)
[comments removed and spacing adjusted for comparison]

```
final class PolicyNodeImpl implements PolicyNode {

    private static final String ANY_POLICY = "2.5.29.32.0";
    private PolicyNodeImpl mParent;
    private HashSet mChildren;
    private String mValidPolicy;
    private HashSet mQualifierSet;
    private boolean mCriticalityIndicator;
    private HashSet mExpectedPolicySet;
    private boolean mOriginalExpectedPolicySet;
    private int mDepth;
    private boolean isImmutable = false;

    PolicyNodeImpl(PolicyNodeImpl parent, String validPolicy,
        Set qualifierSet,boolean criticalityIndicator,
        Set expectedPolicySet,boolean generatedByPolicyMapping) {

        mParent = parent;
        mChildren = new HashSet();
        if (validPolicy != null)
            mValidPolicy = validPolicy;
        else
            mValidPolicy = "";

        if (qualifierSet != null)
            mQualifierSet = new HashSet(qualifierSet);
        else
            mQualifierSet = new HashSet();

        mCriticalityIndicator = criticalityIndicator;
        if (expectedPolicySet != null)
            mExpectedPolicySet = new HashSet(expectedPolic
```

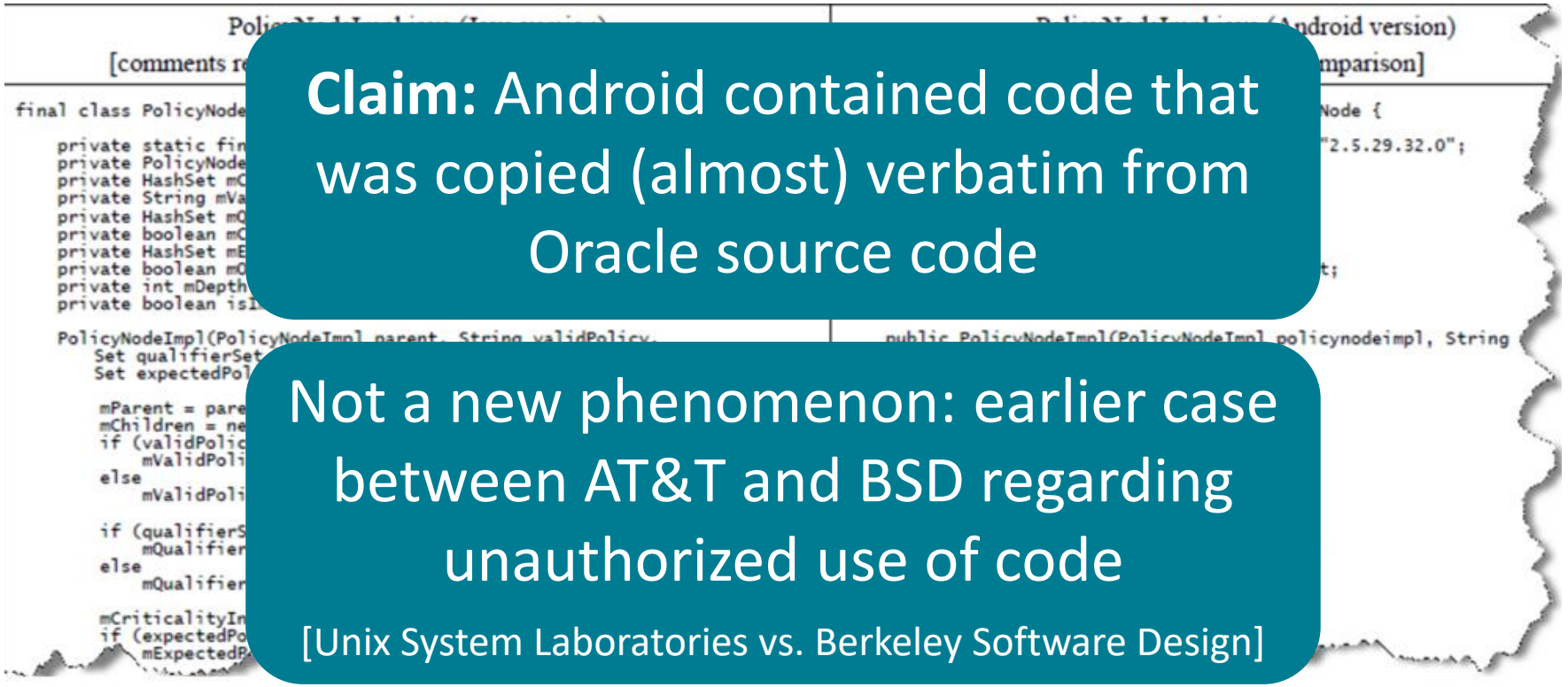
PolicyNodeImpl.java (Android version)
[spacing adjusted for comparison]

```
public class PolicyNodeImpl implements PolicyNode {

    private static final String ANY_POLICY = "2.5.29.32.0";
    private PolicyNodeImpl mParent;
    private HashSet mChildren;
    private String mValidPolicy;
    private HashSet mQualifierSet;
    private boolean mCriticalityIndicator;
    private HashSet mExpectedPolicySet;
    private boolean mOriginalExpectedPolicySet;
    private int mDepth;
    private boolean isImmutable;

    public PolicyNodeImpl(PolicyNodeImpl policynodeimpl, String
        Set set, boolean flag,
        Set set1, boolean flag1) {
        isImmutable = false;
        mParent = policynodeimpl;
        mChildren = new HashSet();
        if(s != null) {
            mValidPolicy = s;
        } else {
            mValidPolicy = "";
        }
        if(set != null) {
            mQualifierSet = new HashSet(set);
        } else {
            mQualifierSet = new HashSet();
        }
        mCriticalityIndicator = flag;
        if(set1 != null) {
            mExpectedPolicySet = new HashSet(
```

Snippet of code used in Oracle copyright and patent dispute against Google

Watermarking Software

How do we prove ownership of software?



Claim: Android contained code that was copied (almost) verbatim from Oracle source code

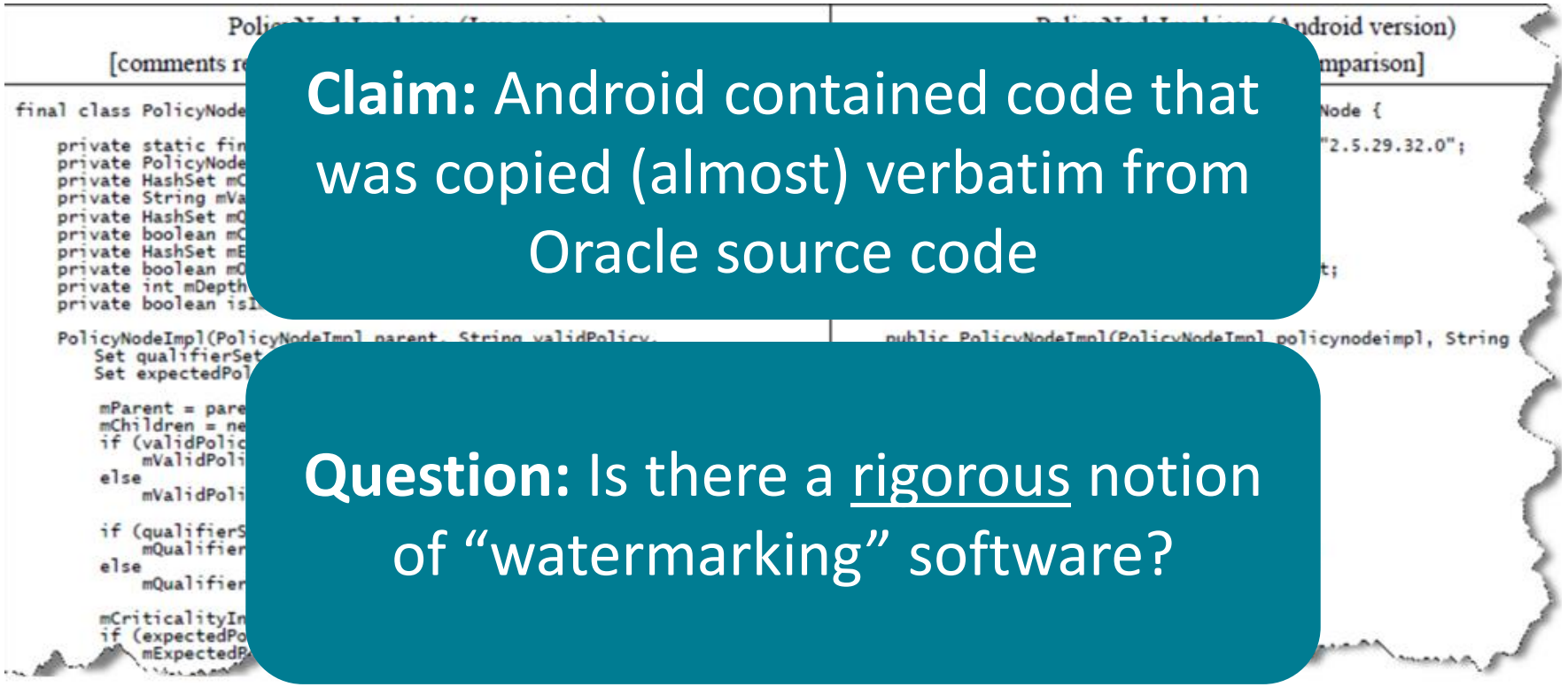
Not a new phenomenon: earlier case between AT&T and BSD regarding unauthorized use of code

[Unix System Laboratories vs. Berkeley Software Design]

Snippet of code used in Oracle copyright and patent dispute against Google

Watermarking Software

How do we prove ownership of software?



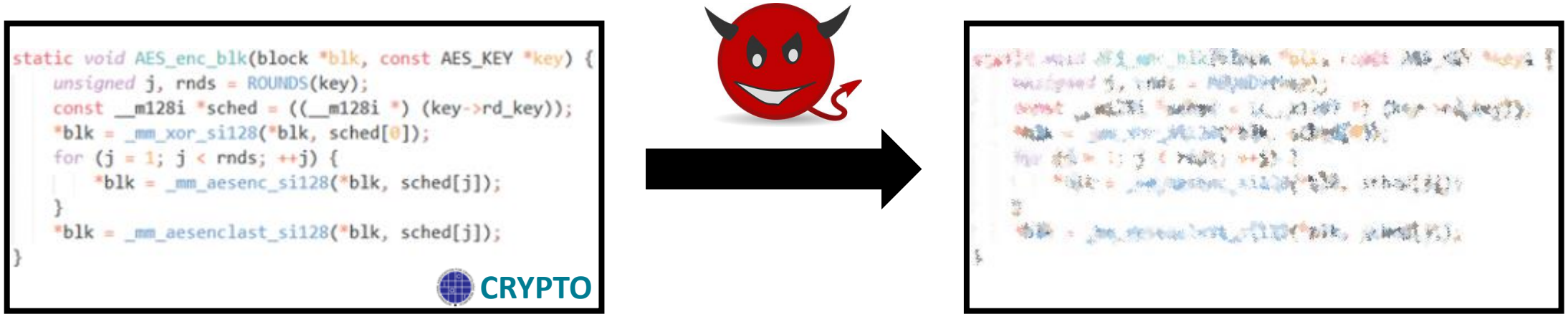
Claim: Android contained code that was copied (almost) verbatim from Oracle source code

Question: Is there a rigorous notion of “watermarking” software?

Snippet of code used in Oracle copyright and patent dispute against Google

Watermarking Software

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]



Embed a "mark" within a program

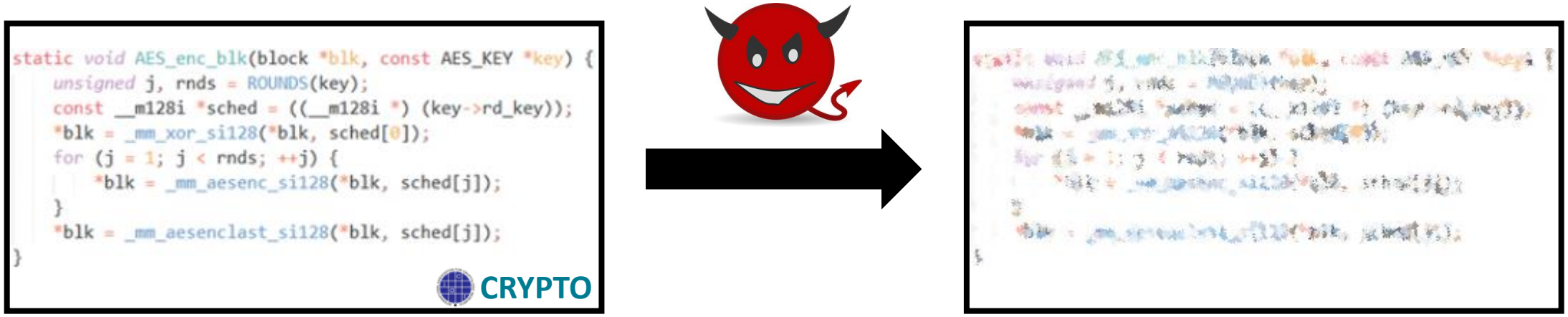
If mark is removed, then program is destroyed

Two main algorithms (simplified):

- $\mathrm{Mark}(C) \to C'$: Takes a circuit C and outputs a marked circuit C'
- $\mathrm{Verify}(C') \to \{0,1\}$: Tests whether a circuit C' is marked or not

Watermarking Software

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]



Embe

Notion extend to setting where watermark can be any string

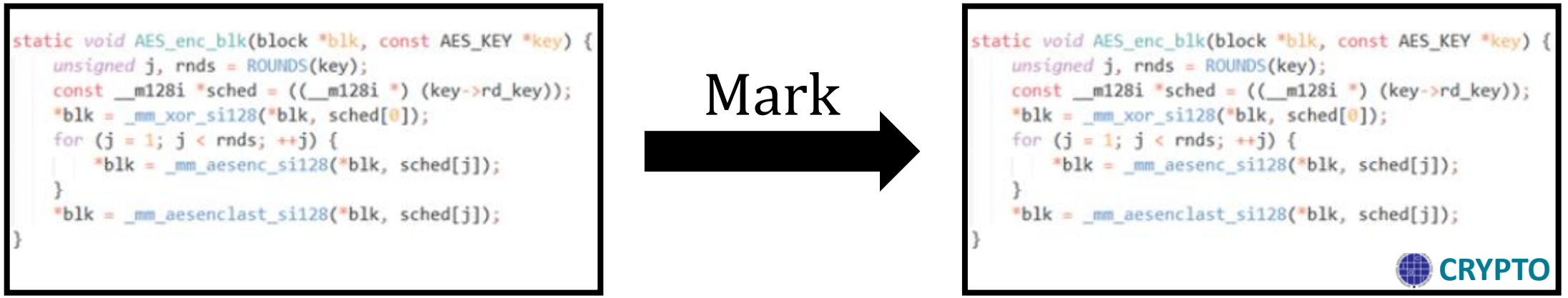
If mark is removed, then program is destroyed

Two main algo[illegible]s (simplified):

- $\mathrm{Mark}(C) \to C'$: Takes a circuit C and outputs a marked circuit C'
- $\mathrm{Verify}(C') \to \{0,1\}$: Tests whether a circuit C' is marked or not

Watermarking Software

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]



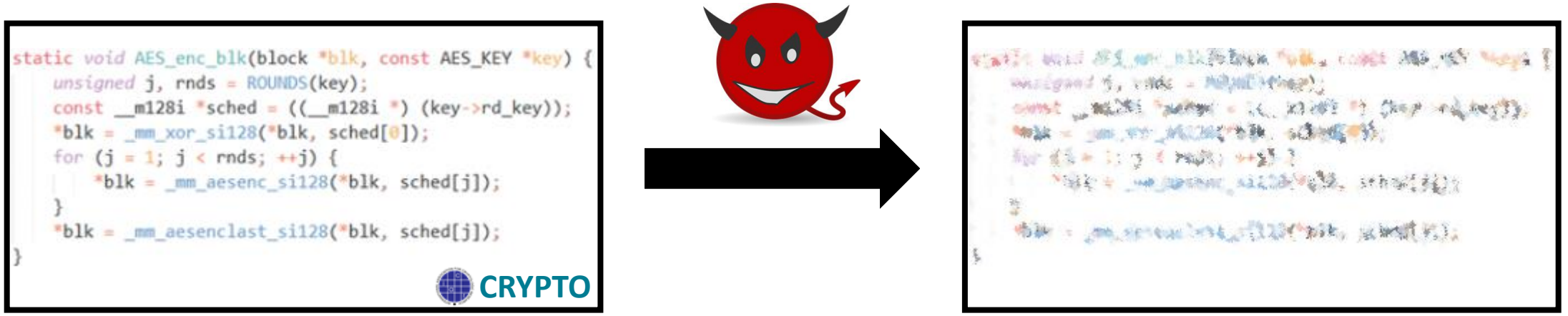
Functionality-preserving: On input a circuit C, the Mark algorithm outputs a circuit C' where

$$C(x) = C'(x)$$

on almost all inputs x

Watermarking Software

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]



Unremovability: Given a marked program C', no efficient adversary can construct a circuit $C^\star$ where

- $C^\star(x) = C'(x)$ on almost all inputs x
- The circuit $C^\star$ is unmarked: $\mathrm{Verify}(C^\star) = 0$

Watermarking Software

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]

Adversary is very powerful: sees the code of the marked program C' and has complete flexibility in crafting $C^\star$

Unremovability: Given a marked program C', no efficient adversary can construct a circuit $C^\star$ where

- $C^\star(x) = C'(x)$ on almost all inputs x
- The circuit $C^\star$ is unmarked: $\mathrm{Verify}(C^\star) = 0$

Watermarking Software

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]

```
static void AES_enc_blk(block *blk, const AES_KEY *key) {
    unsigned j, rnds = ROUNDS(key);
    const __m128i *sched = ((__m128i *) (key->rd_key));
    *blk = _mm_xor_si128(*blk, sched[0]);
    for (j = 1; j < rnds; ++j) {
        *blk = _mm_aesenc_si128(*blk, sched[j]);
    }
    *blk = _mm_aesenclast_si128(*blk, sched[j]);
}
```

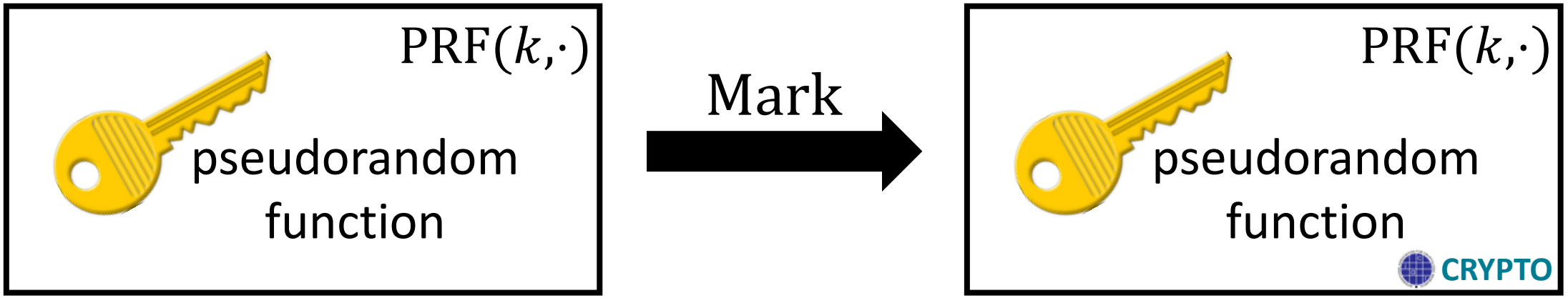
CRYPTO

Learning the original (unmarked) function gives a way to remove the watermark

- Notion only achievable for functions that are not learnable
- Focus has been on cryptographic functions

Watermarking Cryptographic Programs

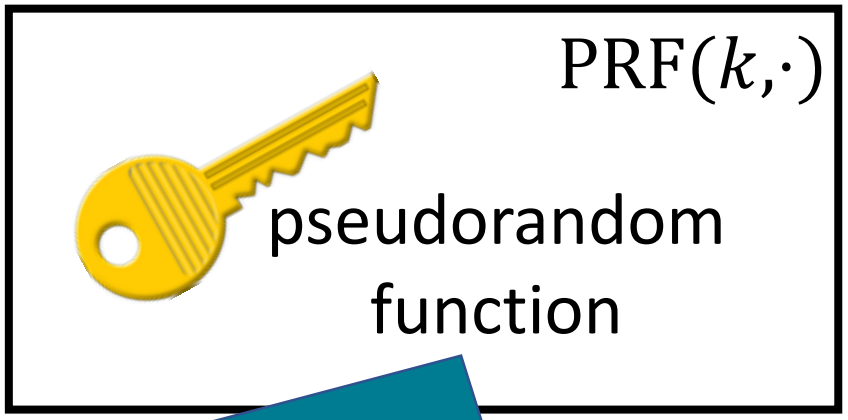
[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]



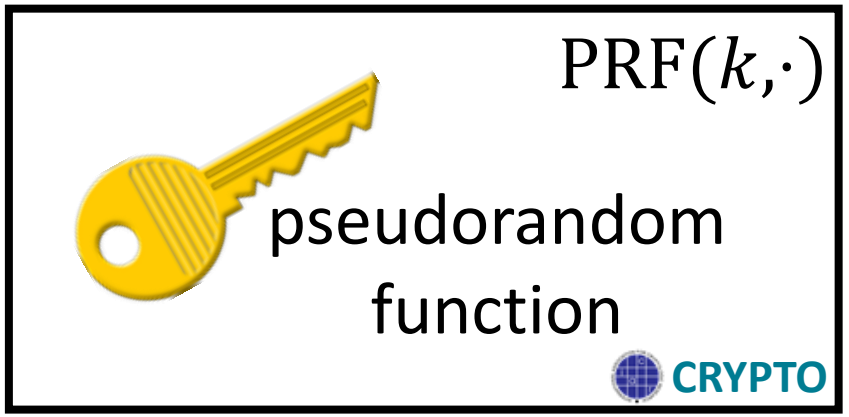
- Focus of this work: watermarking PRFs [CHNVW16, BLW17, KW17]

Watermarking Cryptographic Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]



Mark

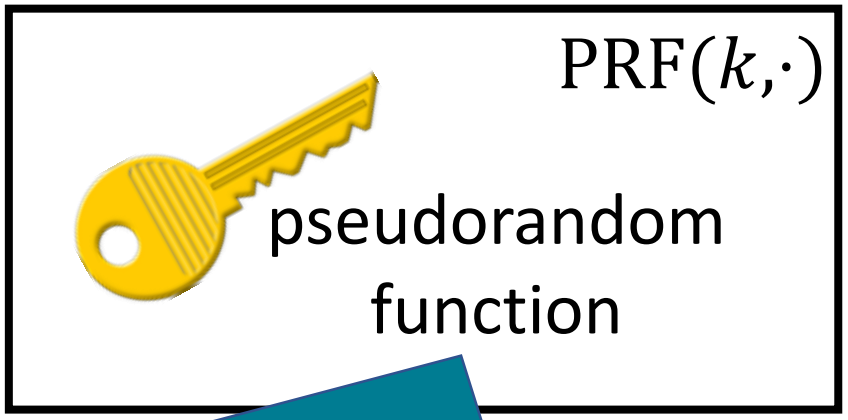


A function whose input-output behavior is unpredictable (looks like a random function) – e.g., AES

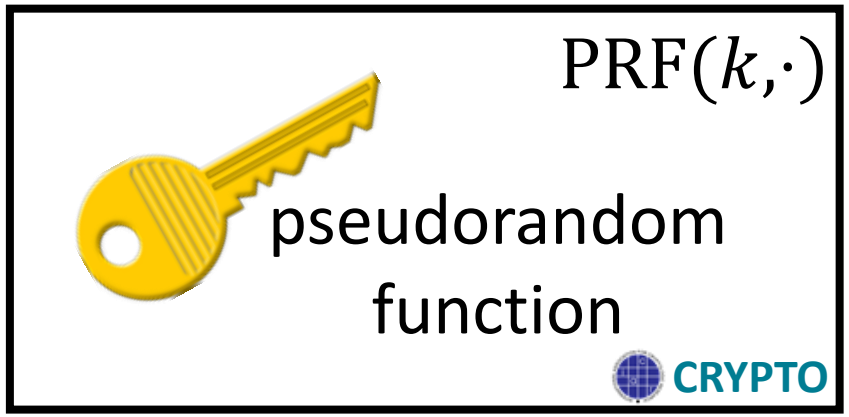
ng PRFs [CHNVW16, BLW17, KW17]

Watermarking Cryptographic Programs

[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]



Mark

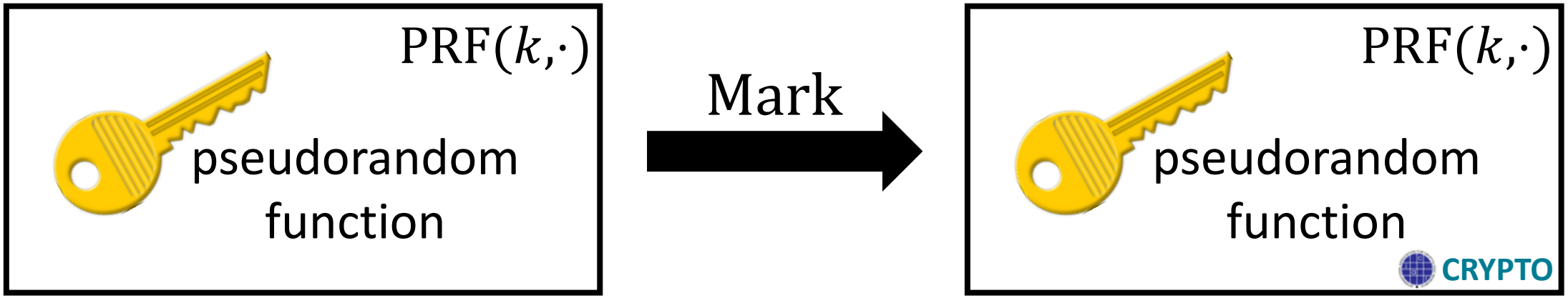


Program has PRF key k hard-wired inside it and on input x, outputs $\mathrm{PRF}(k,x)$

ng PRFs [CHNVW16, BLW17, KW17]

Watermarking Cryptographic Programs

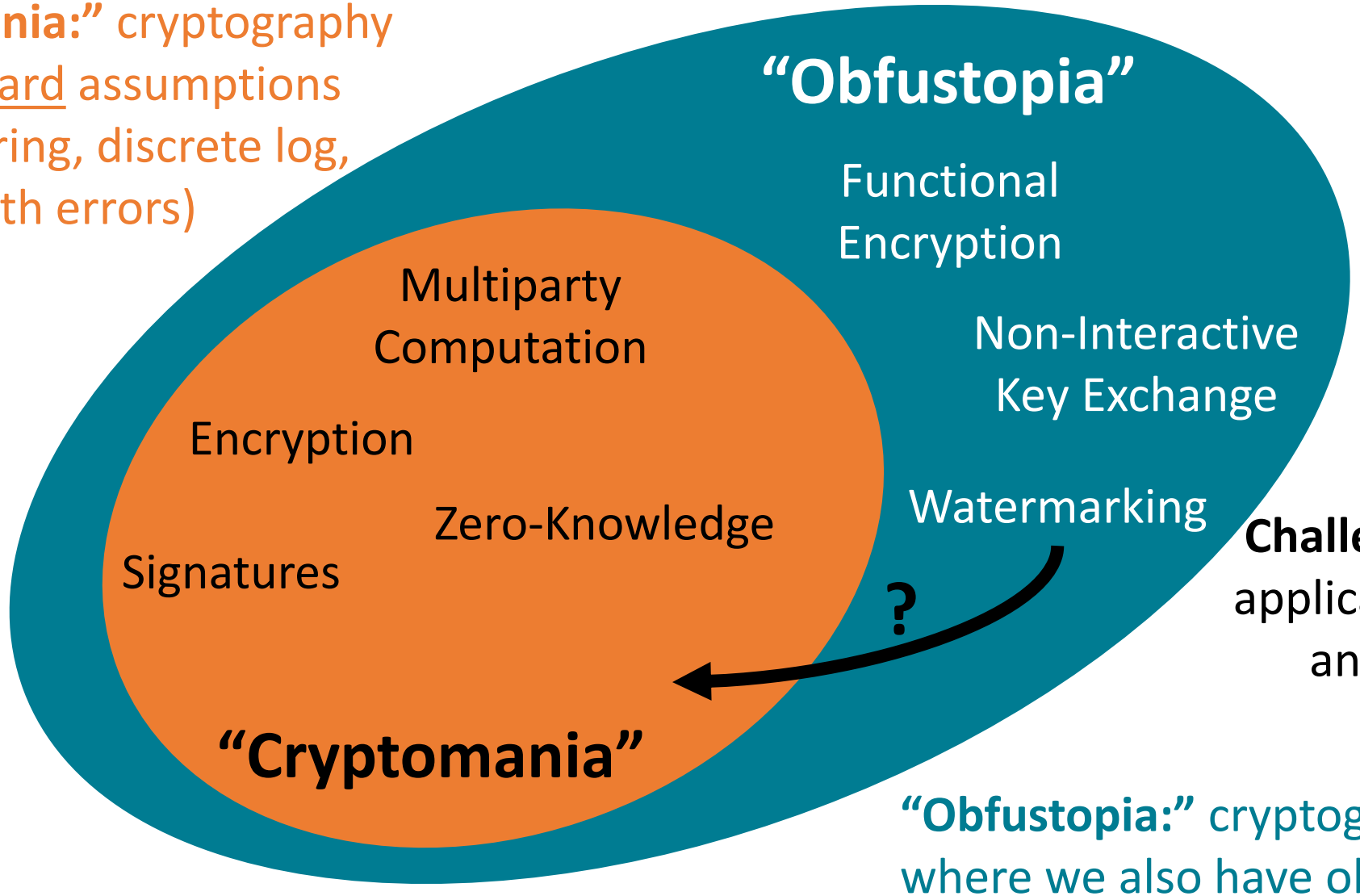
[NSS99, BGIRSVY01, HMW07, YF11, Nis13, CHNVW16, BLW17, KW17]



- Focus of this work: watermarking PRFs [CHNVW16, BLW17, KW17]
- Enables watermarking of symmetric primitives built from PRFs (e.g., encryption, message authentication codes)
- **Goal:** build watermarking from standard and implementable assumptions

Brief Digression: The Landscape of Cryptography

"Cryptomania:" cryptography from standard assumptions (e.g., factoring, discrete log, learning with errors)



Challenge: realizing these applications from standard and implementable assumptions

"Obfustopia:" cryptography where we also have obfuscation

Key Notion: Private Translucent PRFs [*CRYPTO* '17]

joint work with Kim (and recipient of Best Young-Researcher Paper Award)

Starting point: puncturable PRF [BW13, BGI13, KPTZ13]

PRF key

$\text{Puncture}_{x^\star}$

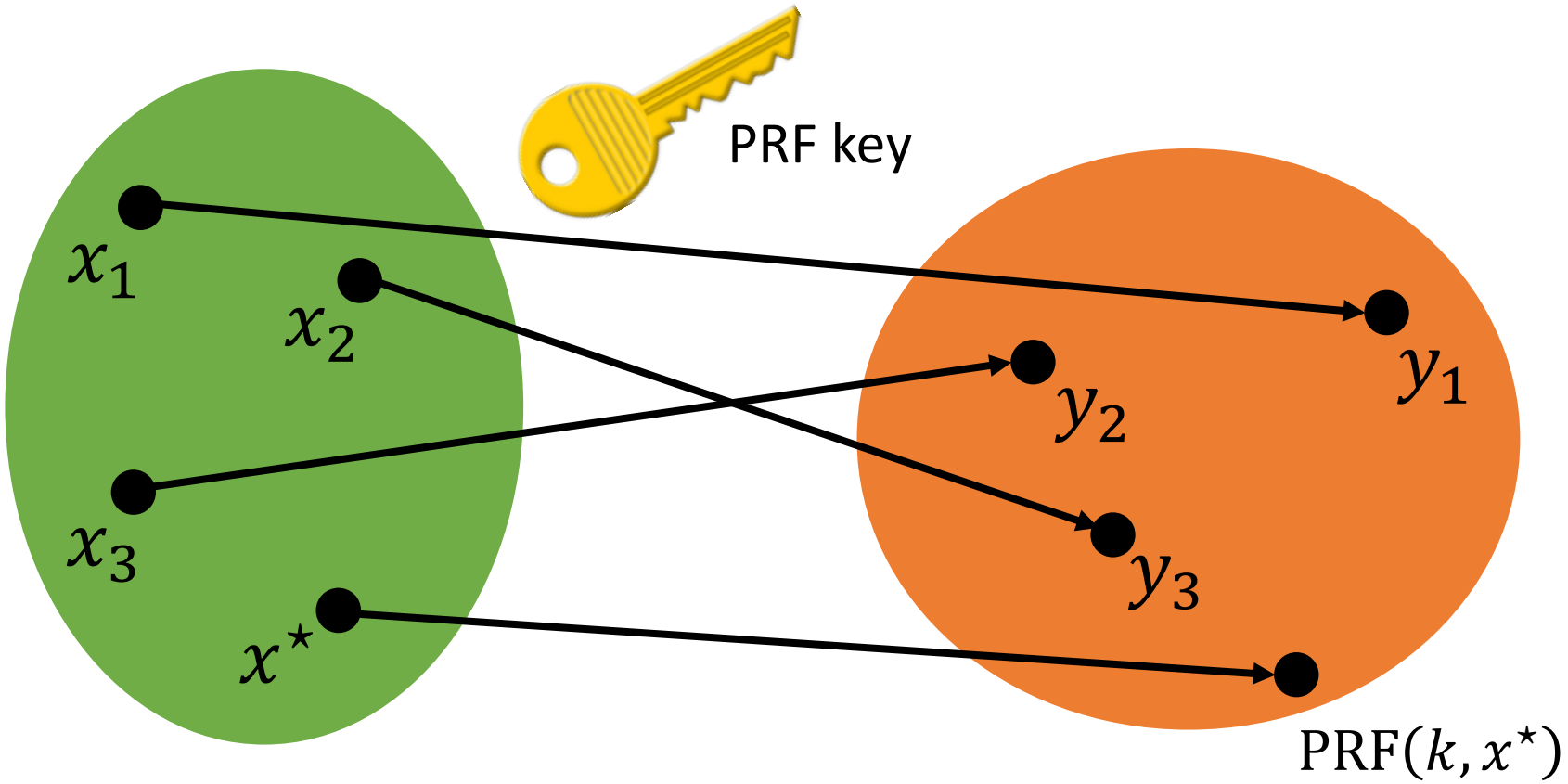
punctured key

Can be used to evaluate the PRF on all points $x \neq x^\star$

Privacy: Punctured key hides $x^\star$

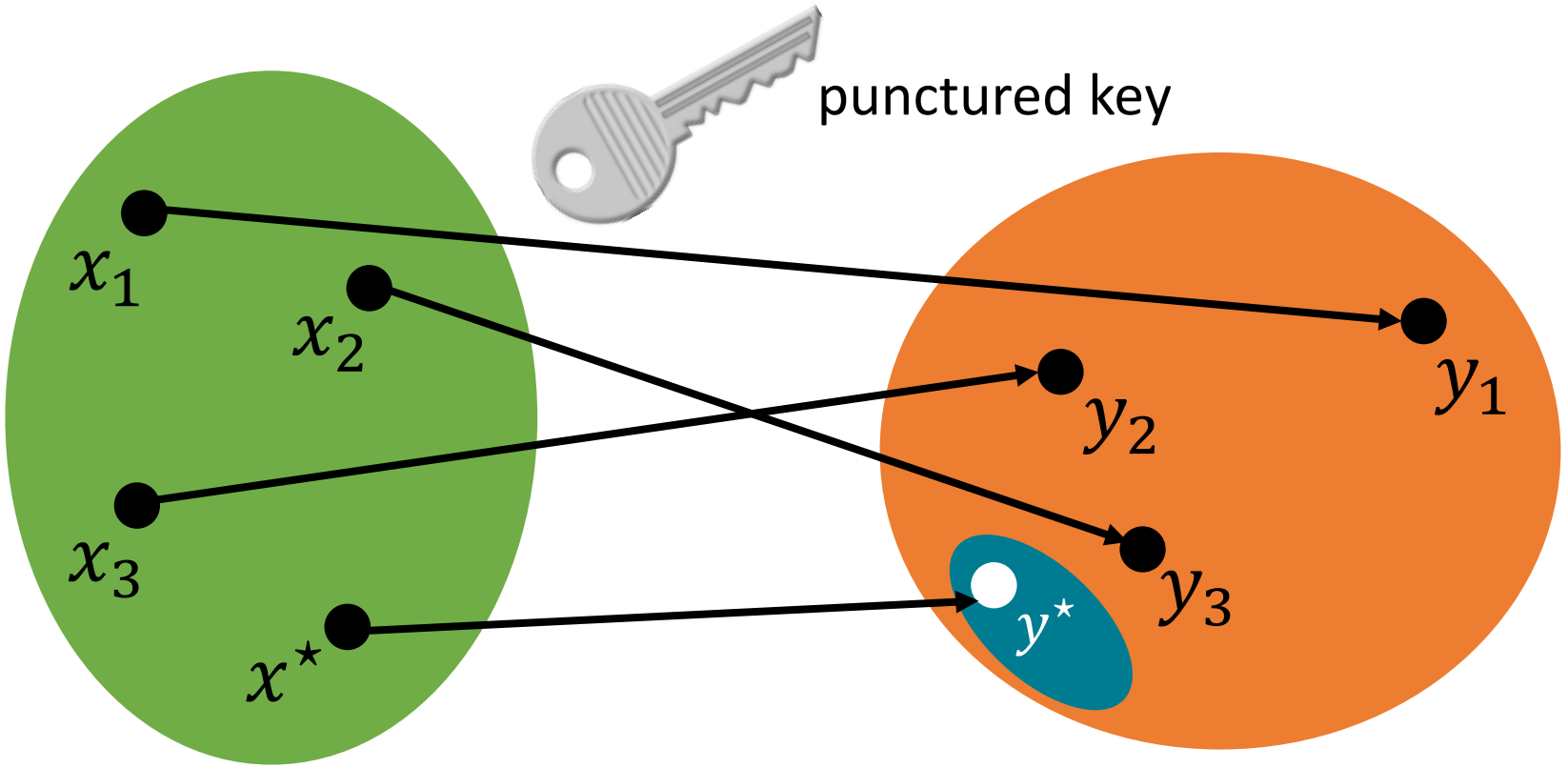
Key Notion: Private Translucent PRFs [*CRYPTO* '17]

joint work with Kim (and recipient of Best Young-Researcher Paper Award)



Key Notion: Private Translucent PRFs [*CRYPTO* '17]

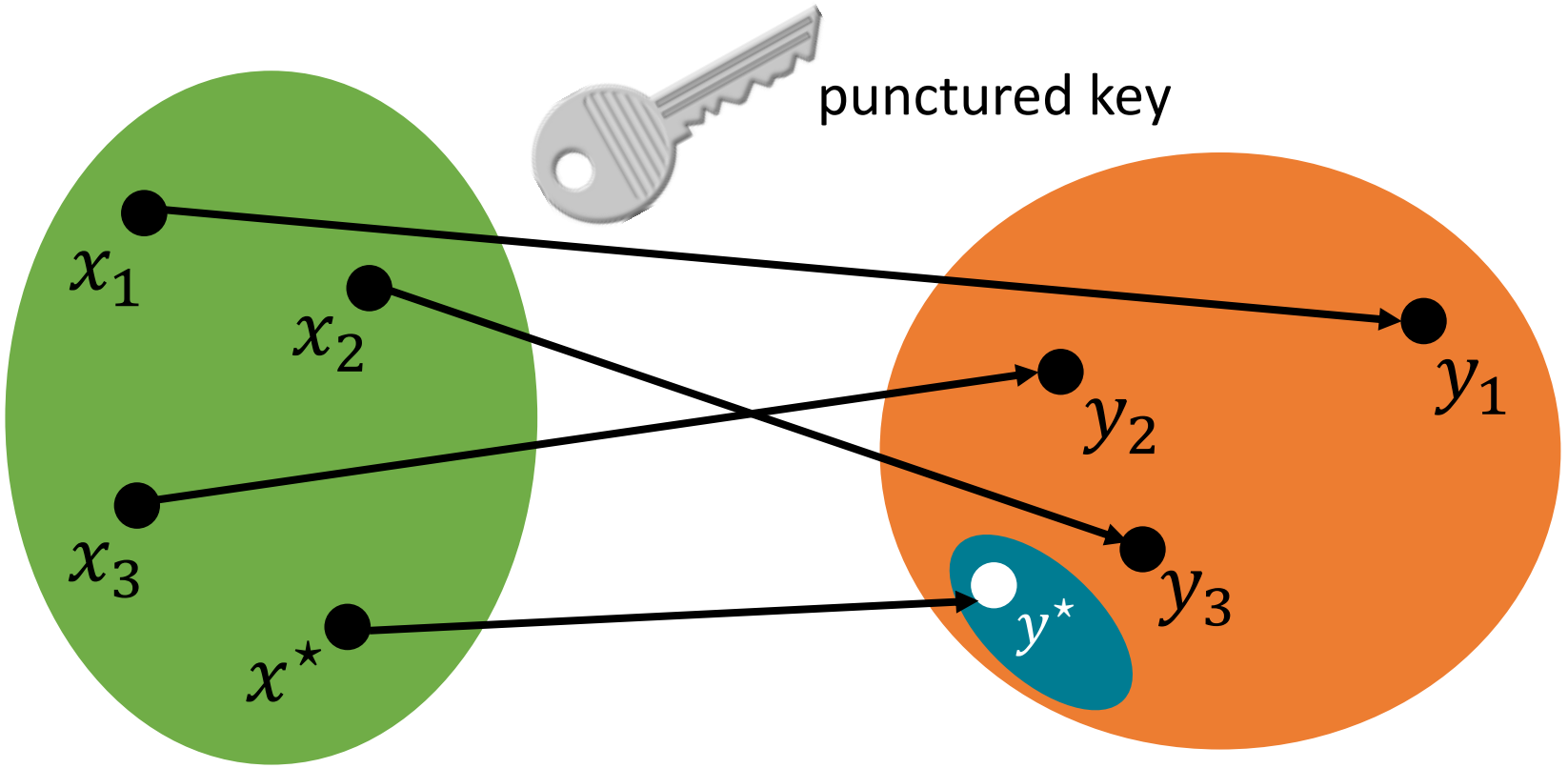
joint work with Kim (and recipient of Best Young-Researcher Paper Award)



Punctured key implements the same function except at $x^\star$

Key Notion: Private Translucent PRFs [*CRYPTO* '17]

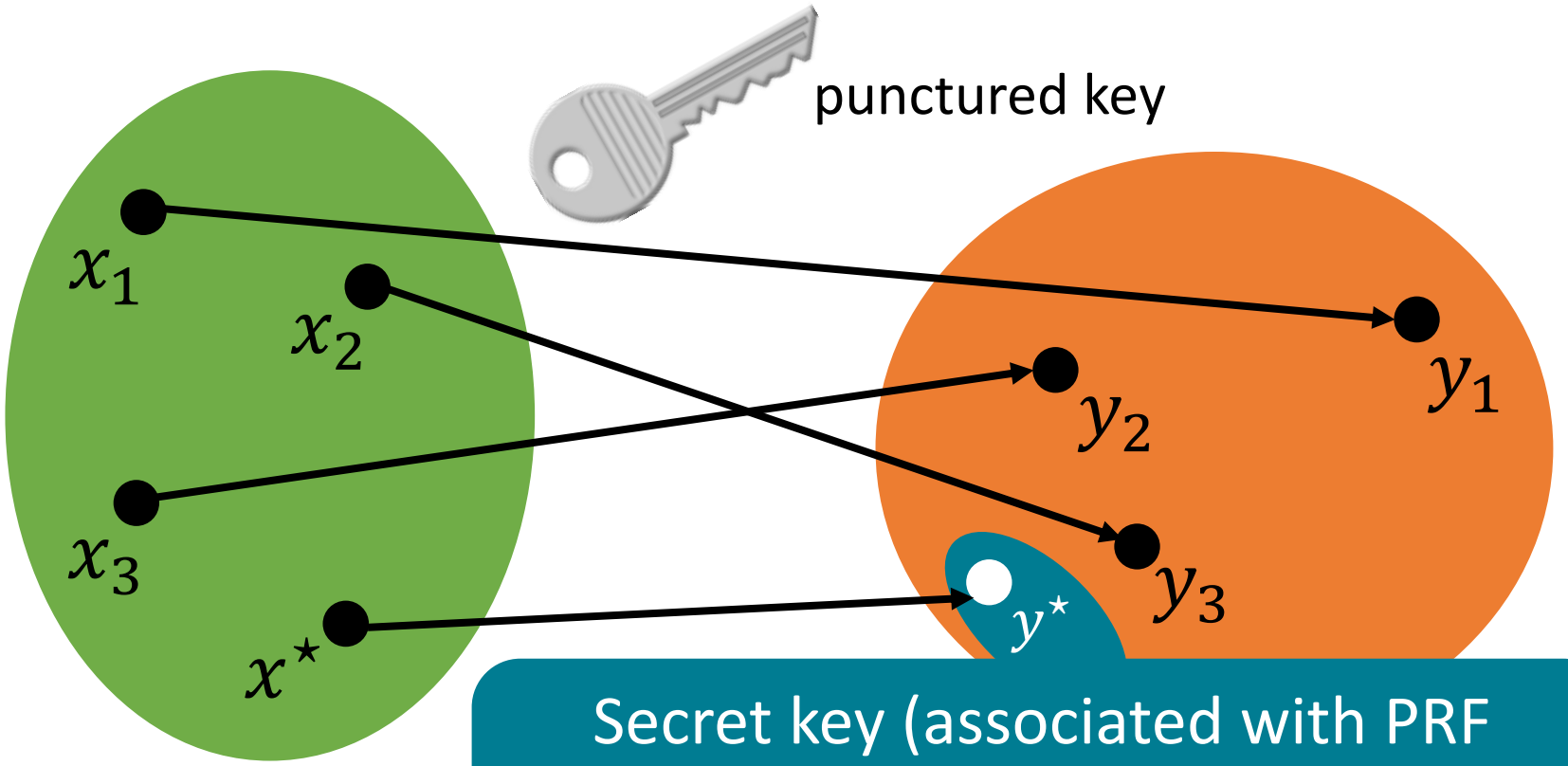
joint work with Kim (and recipient of Best Young-Researcher Paper Award)



Translucent PRF: When punctured key is used to evaluate at $x^\star$, output lies in a sparse, hidden subset of the range

Key Notion: Private Translucent PRFs [*CRYPTO* '17]

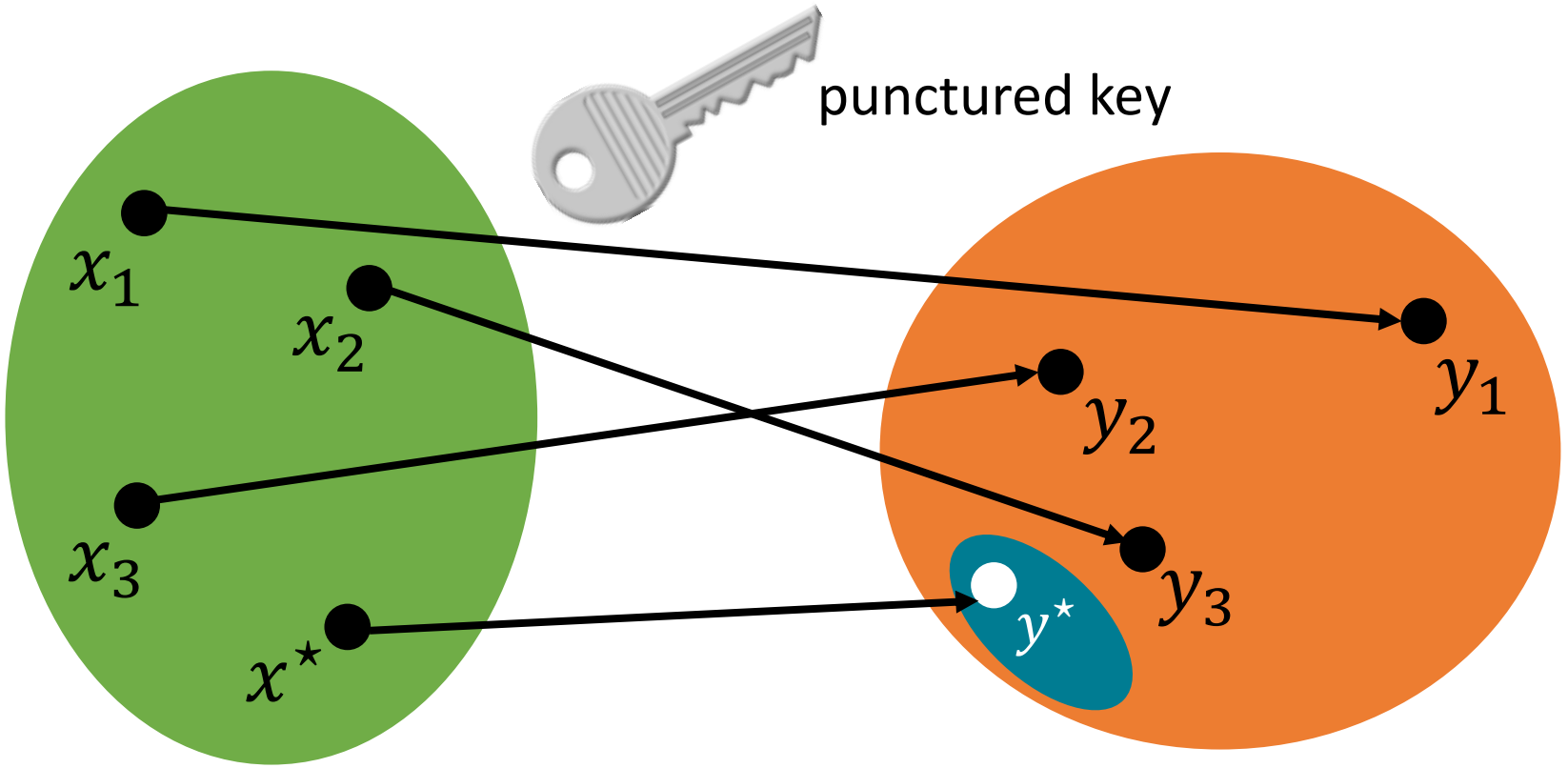
joint work with Kim (and recipient of Best Young-Researcher Paper Award)



Translucent PRF: Whe[illegible]*, output lies in a sparse, hidden subset of the range

Key Notion: Private Translucent PRFs [*CRYPTO* '17]

joint work with Kim (and recipient of Best Young-Researcher Paper Award)

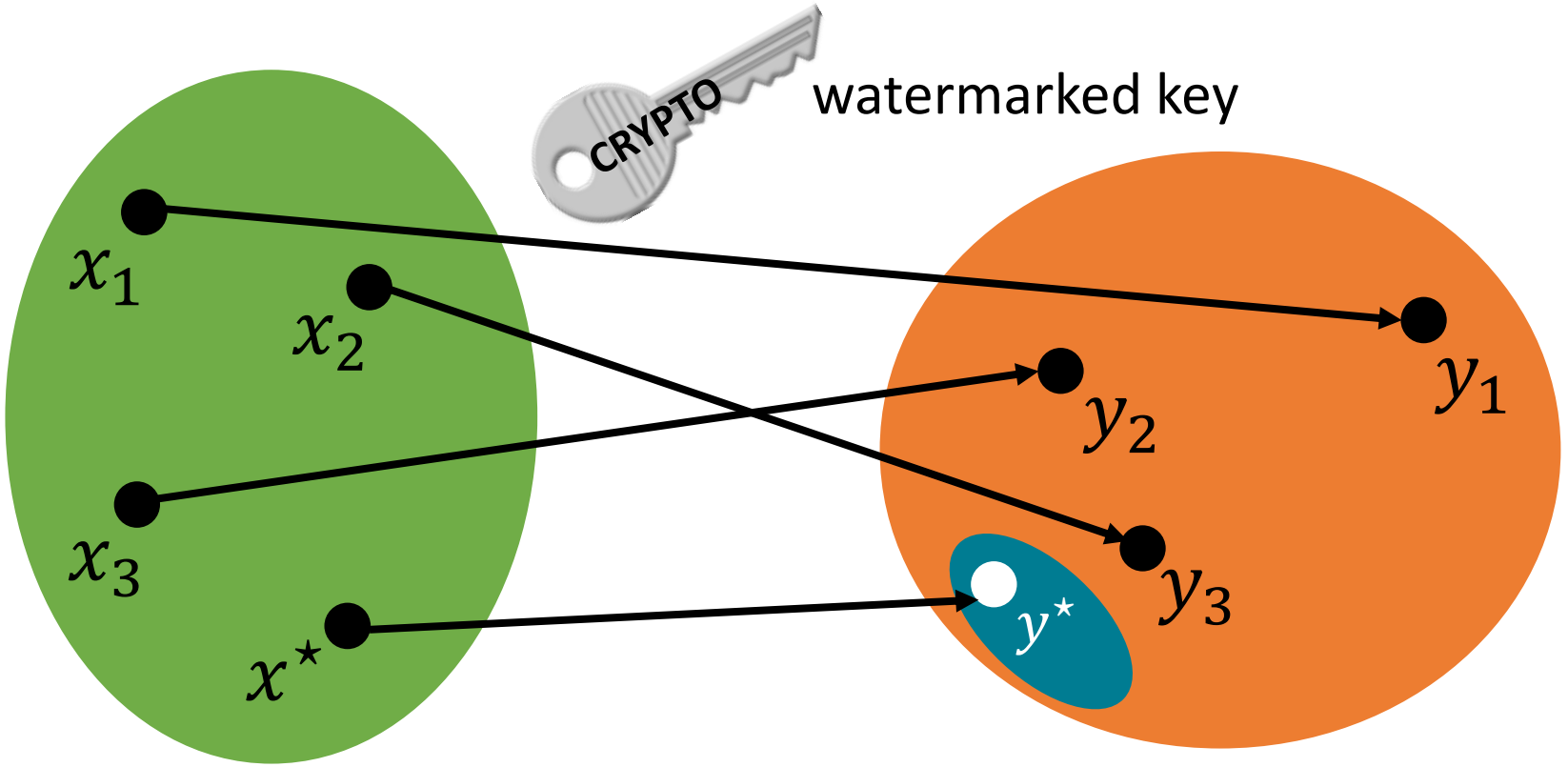


Sets satisfying such properties are called *translucent* [CDNO97]

Values in special set looks indistinguishable from a random value (without secret testing key)

Watermarking from Private Translucent PRFs [*CRYPTO* '17]

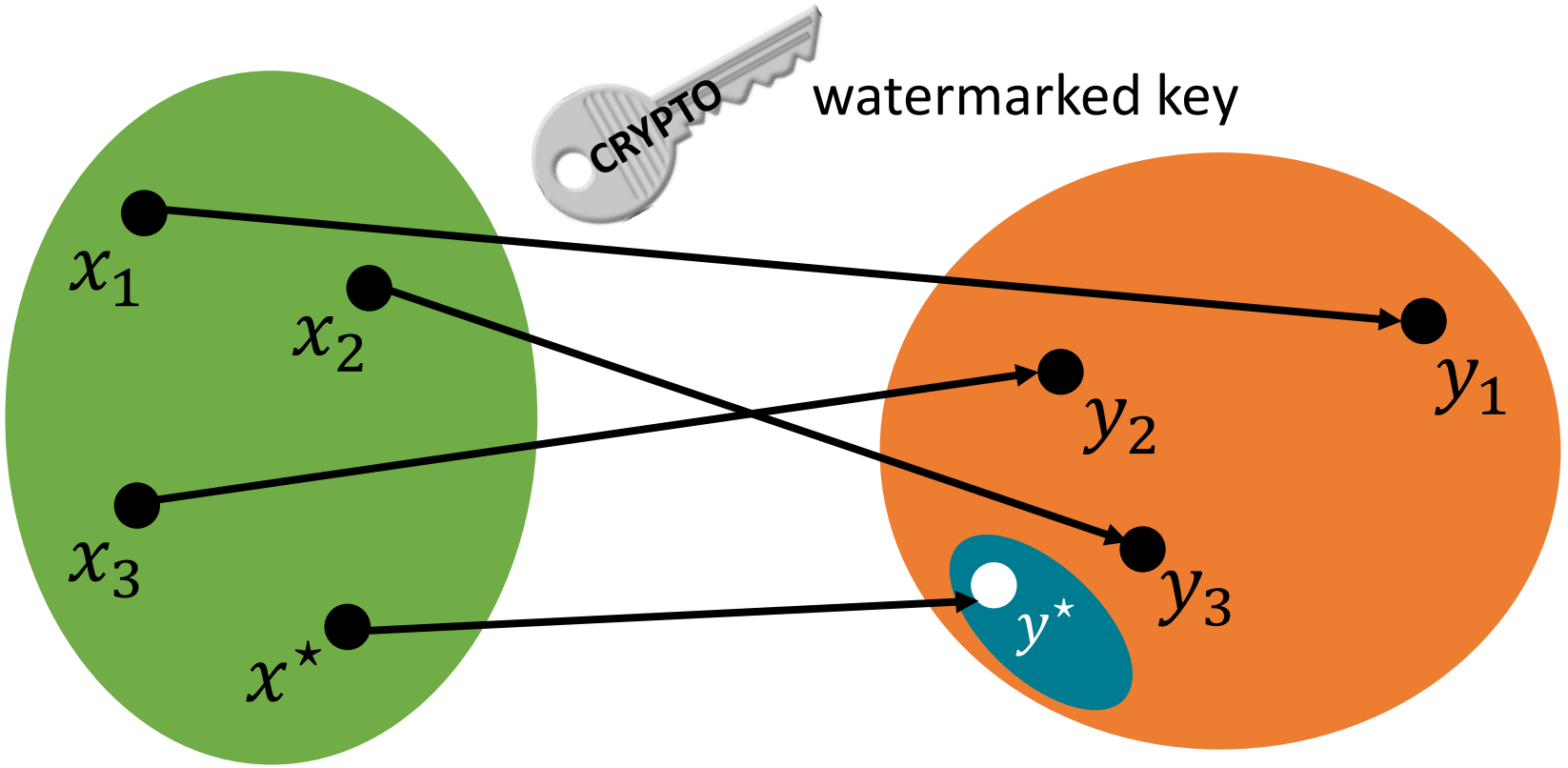
joint work with Kim (and recipient of Best Young-Researcher Paper Award)



Watermarked program just implements evaluation using punctured key (for the private translucent PRF)

Watermarking from Private Translucent PRFs [*CRYPTO* '17]

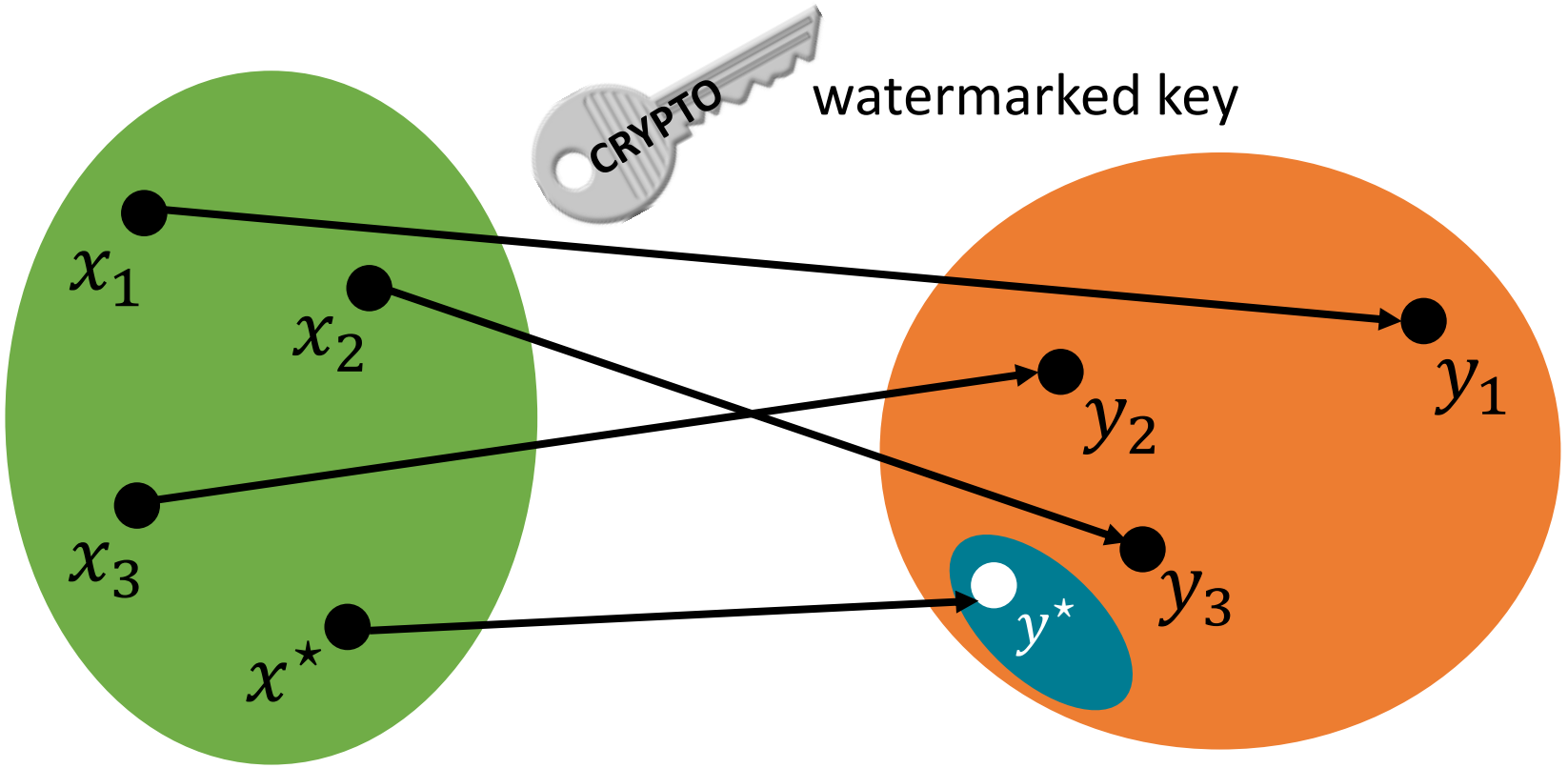
joint work with Kim (and recipient of Best Young-Researcher Paper Award)



Verification: to test whether a program C' is watermarked, check whether $C'(x^\star)$ is in the translucent set (using the testing key for the private translucent PRF)

Watermarking from Private Translucent PRFs [*CRYPTO* '17]

joint work with Kim (and recipient of Best Young-Researcher Paper Award)

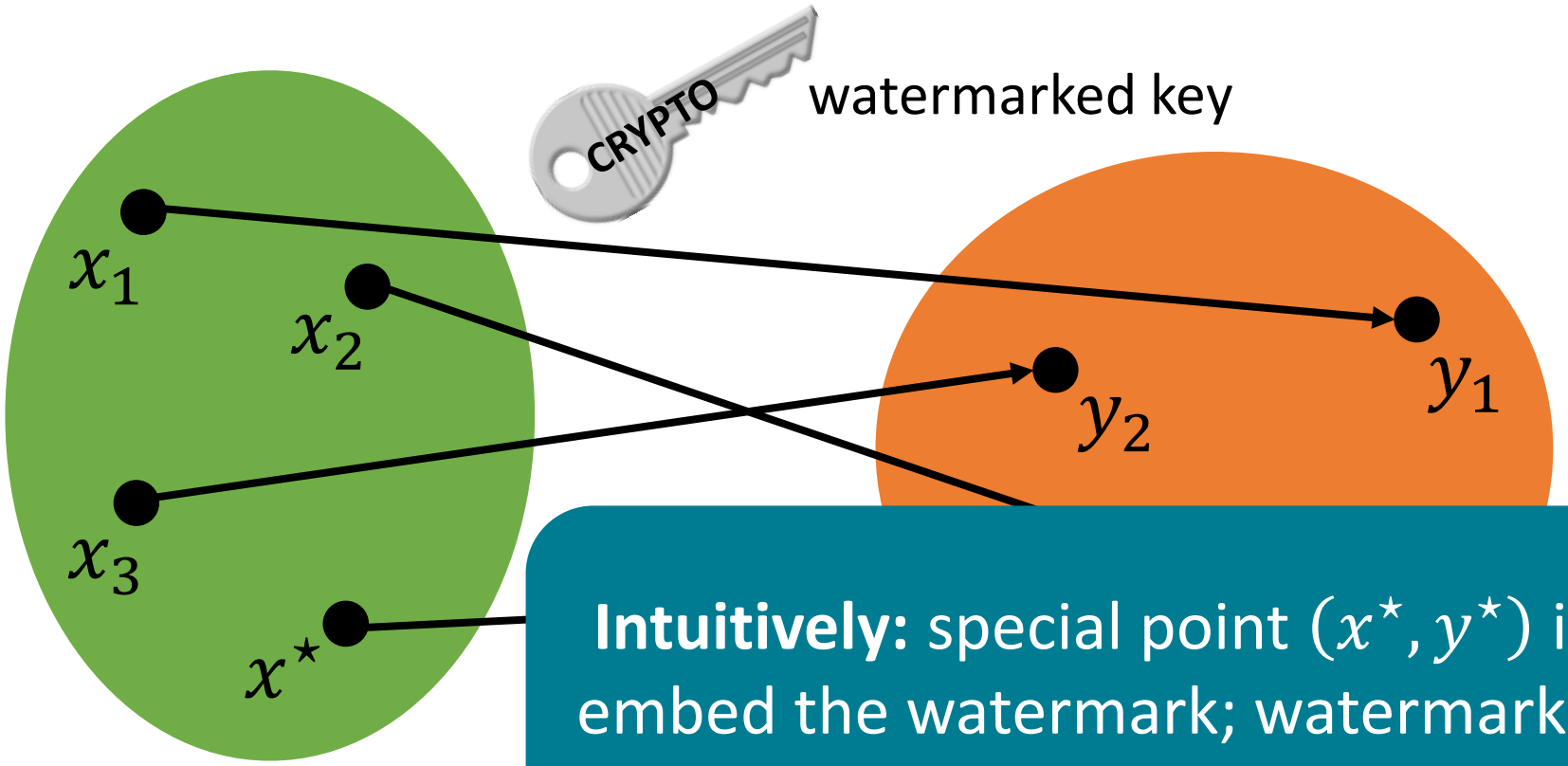


☑ **Functionality-preserving:** function differs at a single point

☑ **Unremovable:** the point $x^\star$ is hidden by privacy, and the value $y^\star$ looks like random element in range by translucency

Watermarking from Private Translucent PRFs [*CRYPTO* '17]

joint work with Kim (and recipient of Best Young-Researcher Paper Award)



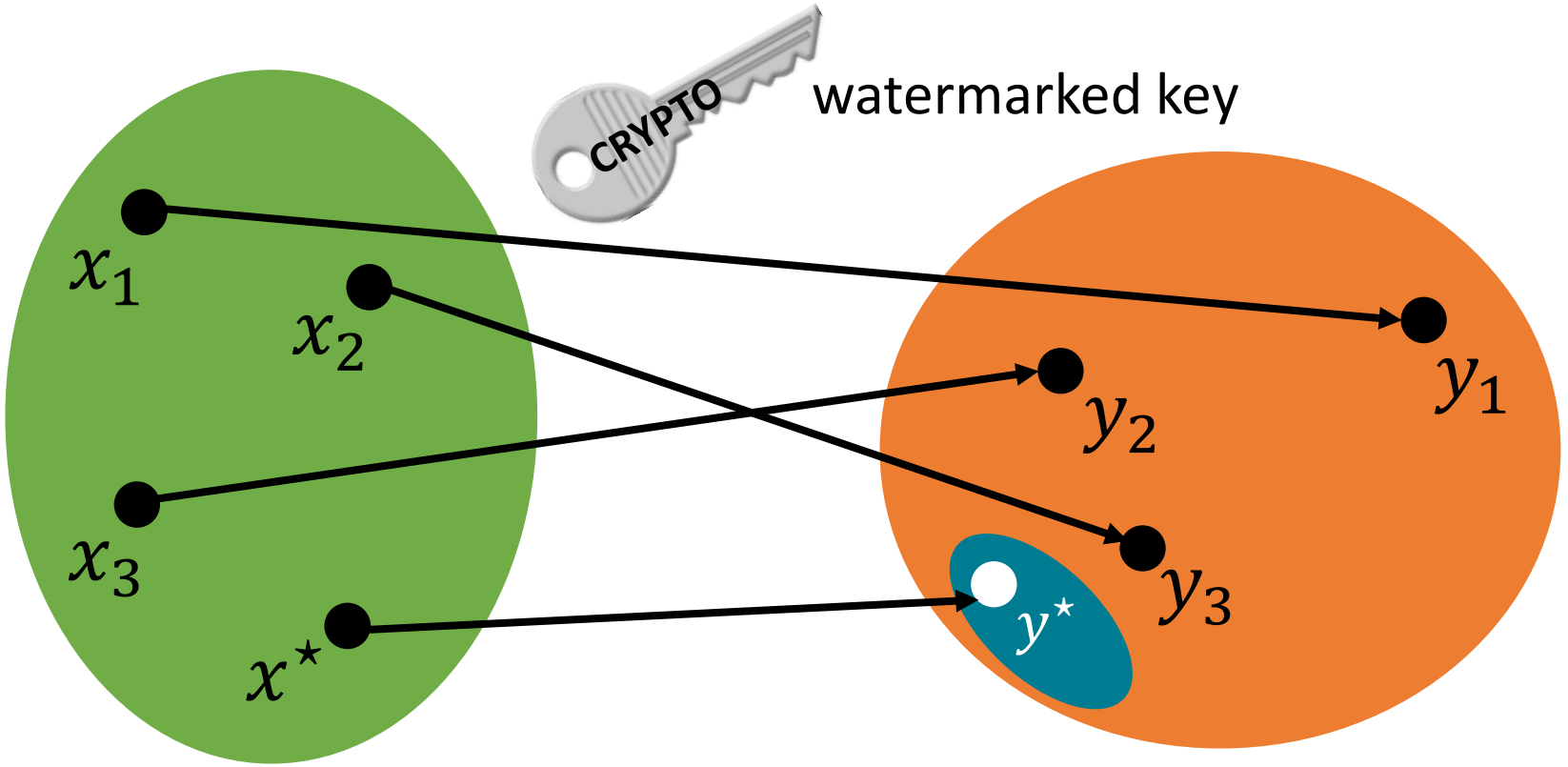
Intuitively: special point $(x^\star, y^\star)$ is used to embed the watermark; watermark is hidden by privacy and translucency properties

☑ **Functionality-preserving:** fur

☑ **Unremovable:** the point $x^\star$ is hidden by privacy, and the value $y^\star$ looks like random element in range by translucency

Watermarking from Private Translucent PRFs [*CRYPTO* '17]

joint work with Kim (and recipient of Best Young-Researcher Paper Award)

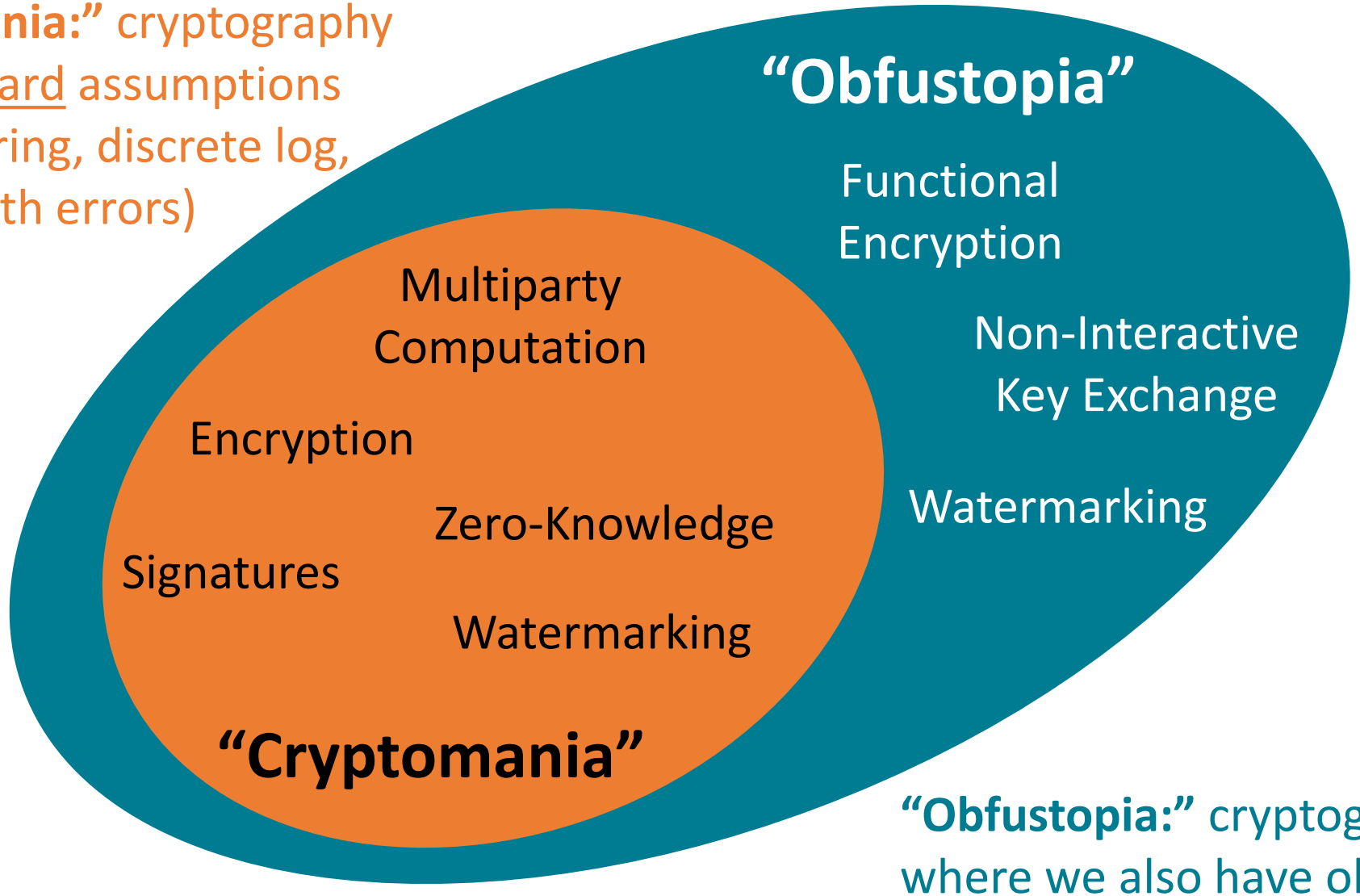


Private translucent PRFs
can be built from standard *lattice* assumptions

Watermarking from Private Translucent PRFs [*CRYPTO* '17]

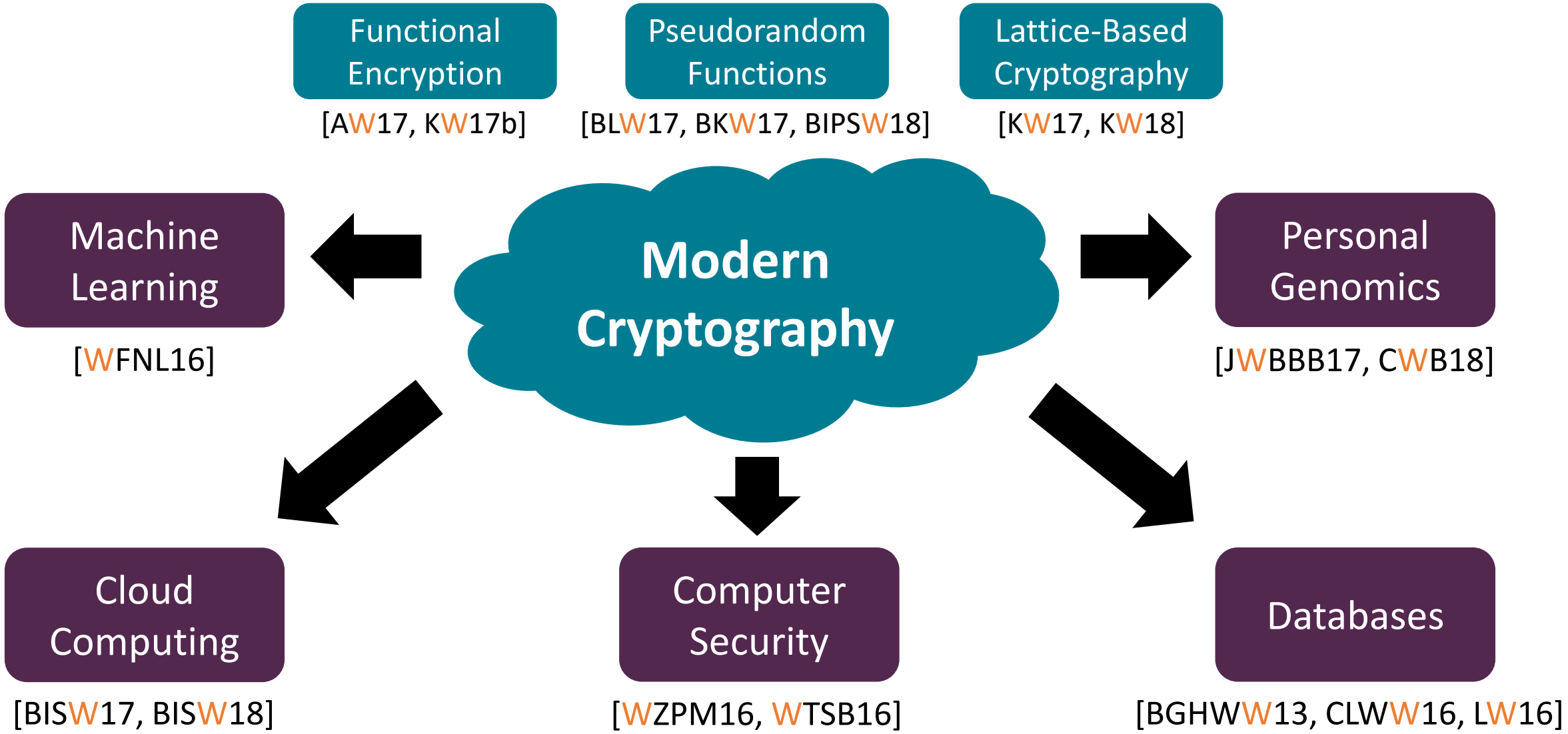
joint work with Kim (and recipient of Best Young-Researcher Paper Award)

"Cryptomania:" cryptography from standard assumptions (e.g., factoring, discrete log, learning with errors)



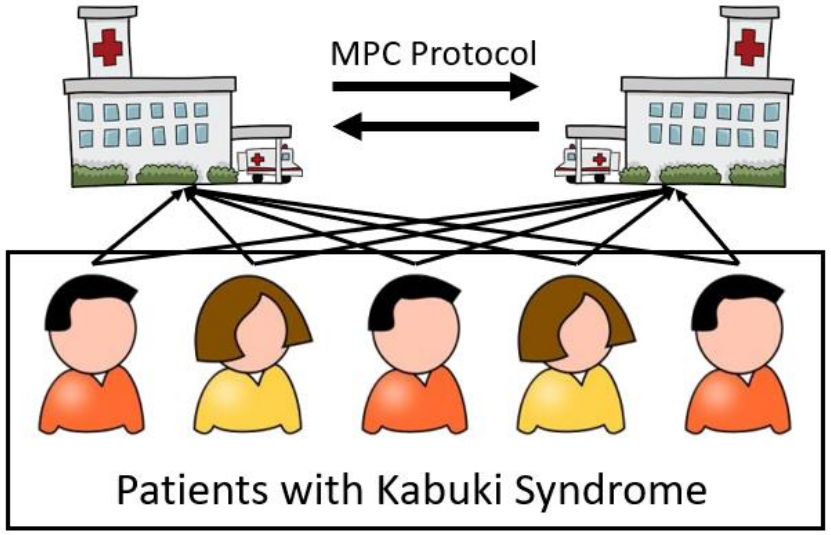
"Obfustopia:" cryptography where we also have obfuscation

My Research from 10,000 Feet

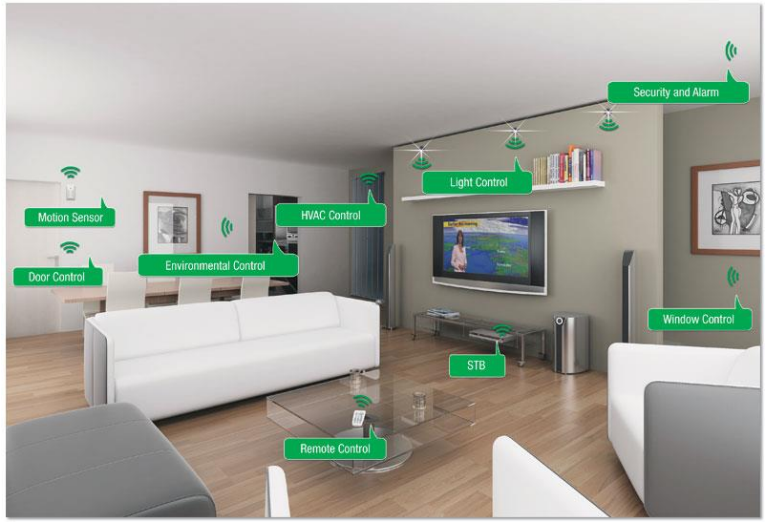


Research Themes and Directions

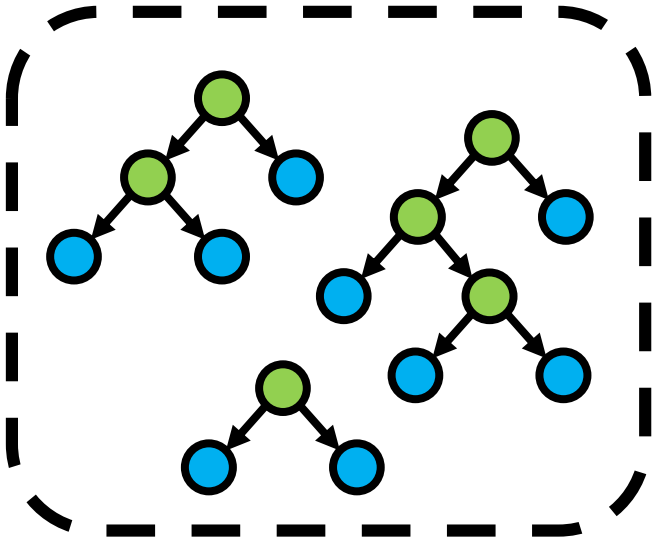
Developing new protocols for privacy-preserving computation



Genome Privacy
[JWBBB17, CWB18]



Internet of Things
[WTSB16]

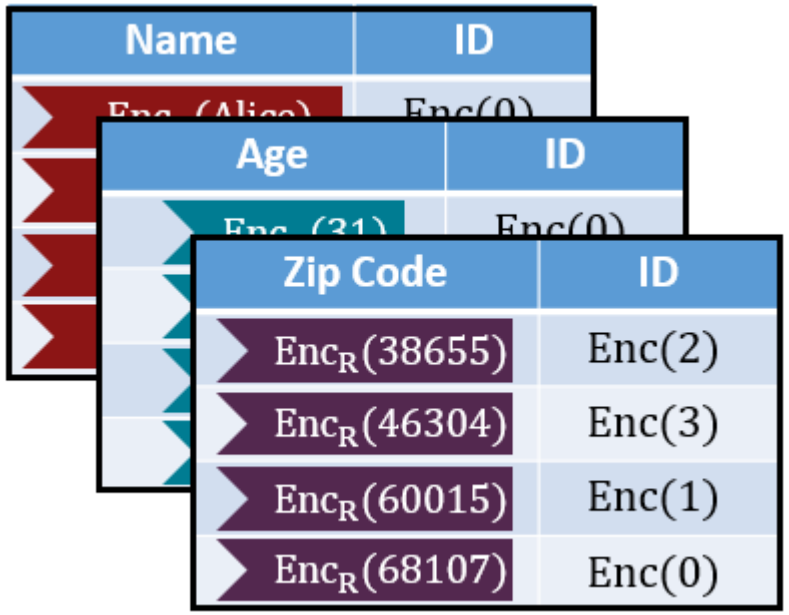


Machine Learning
[WFNL16]

Can we build general frameworks to enable scalable privacy-preserving computation across domains?

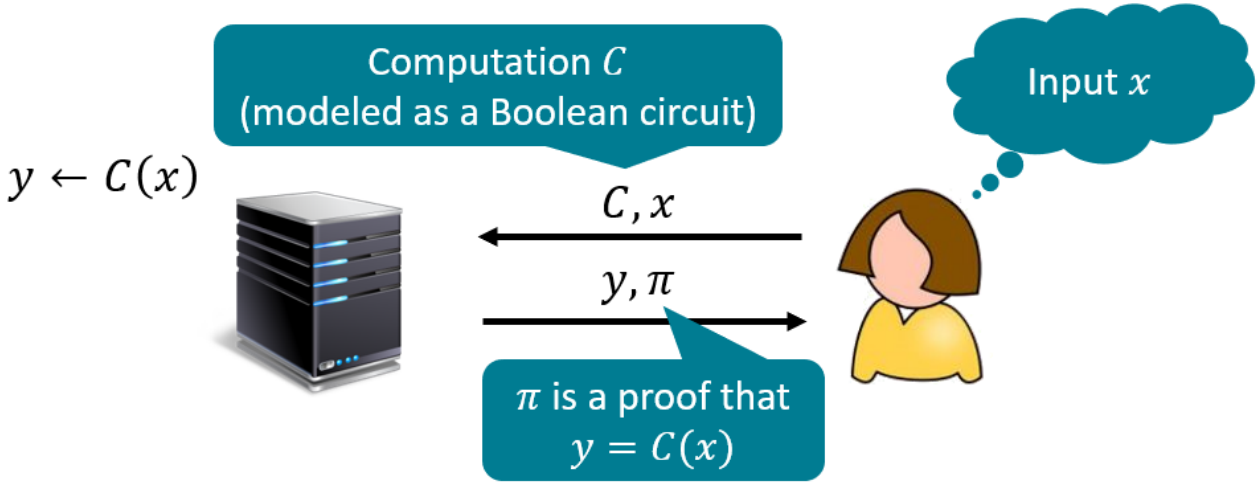
Research Themes and Directions

Build new cryptographic primitives that enable more <u>secure</u> systems



Order-revealing encryption for searching on encrypted data

[CLWW16, LW16]

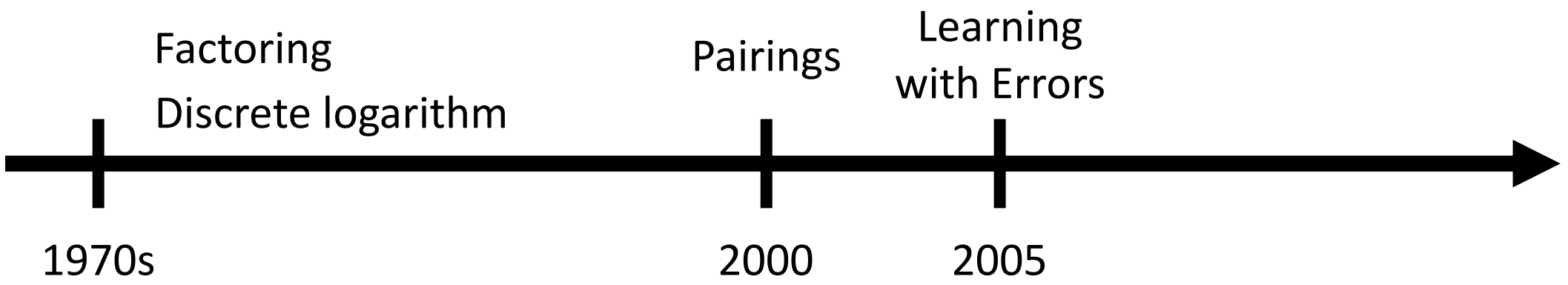


Succinct arguments for verifiable computation

[BISW17, BISW18]

Research Themes and Directions

Realizing complex functionalities from simple assumptions



What new functionalities are possible from standard (and implementable) assumptions?

Thank you!