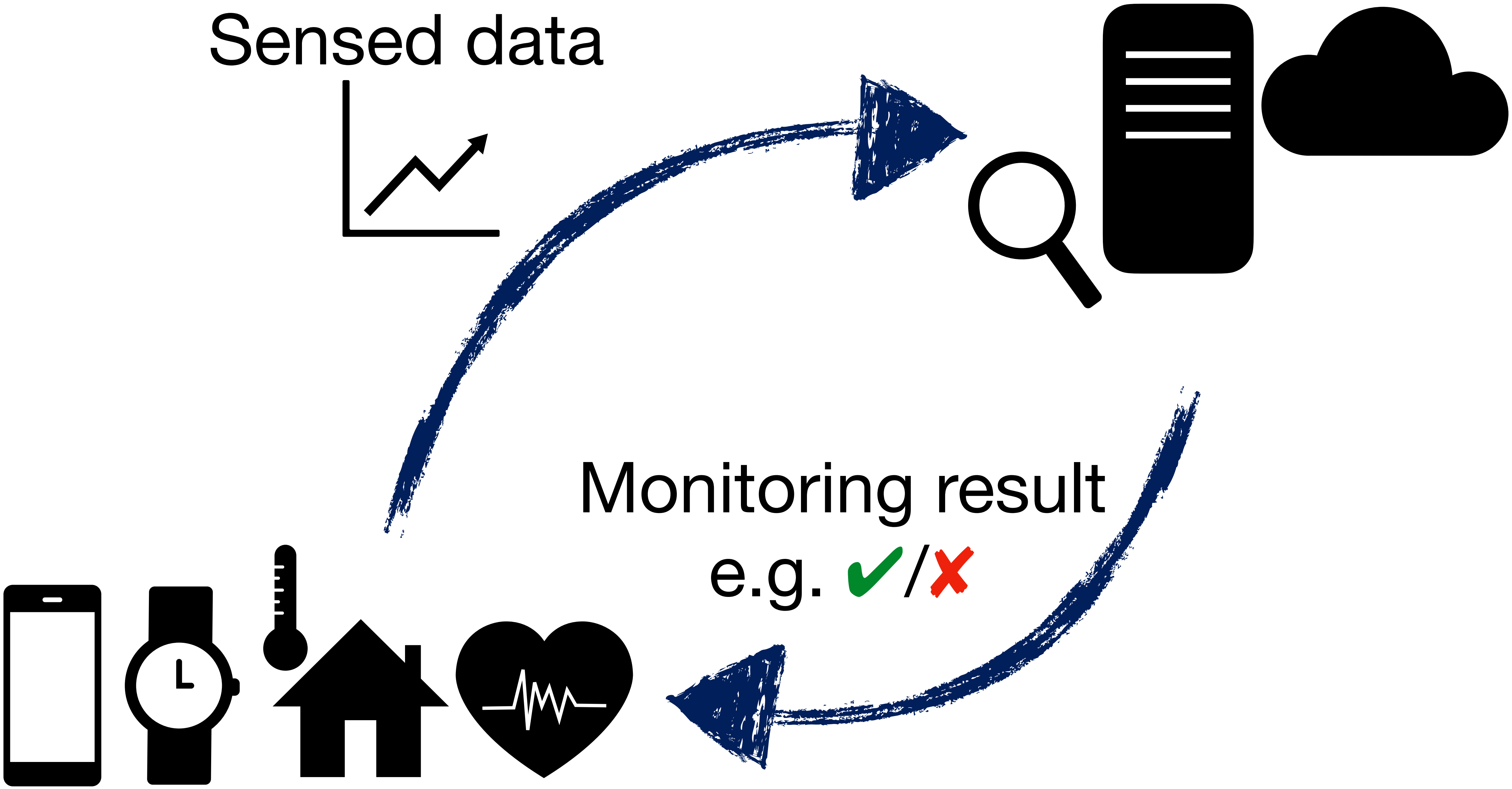
Oblivious Monitoring for Discrete-Time STL via Fully Homomorphic Encryption

Masaki Waga[1] ,Kotaro Matsuoka[1], Takashi Suwa[1], Naoki Matsumoto[1], Ryotaro Banno[2], Song Bian[3], and Kohei Suenaga[1]

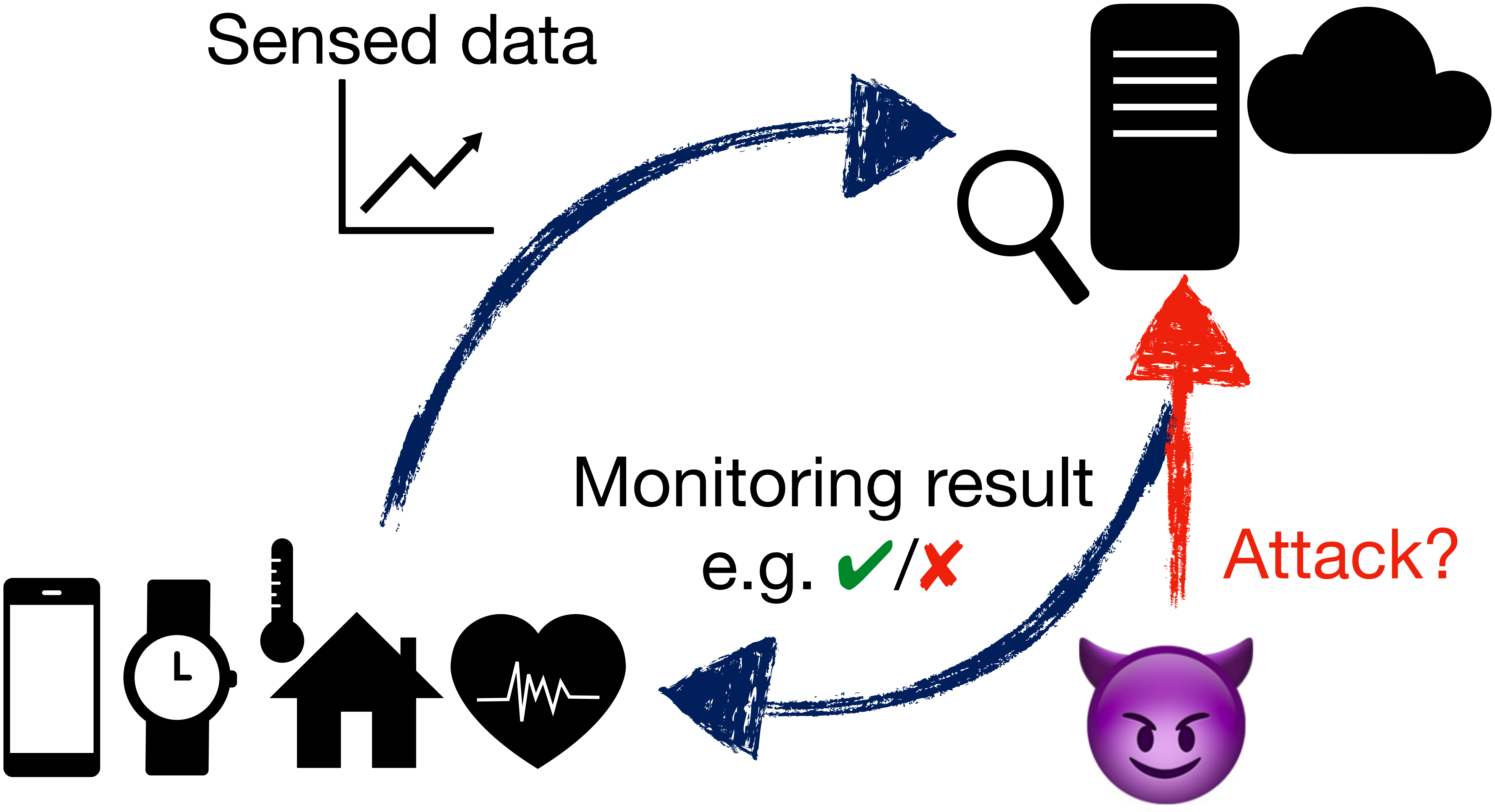
Kyoto University[1], Cybozu, Inc.[2], Beihang University[3]

RV 2024, 15th Oct. 2024

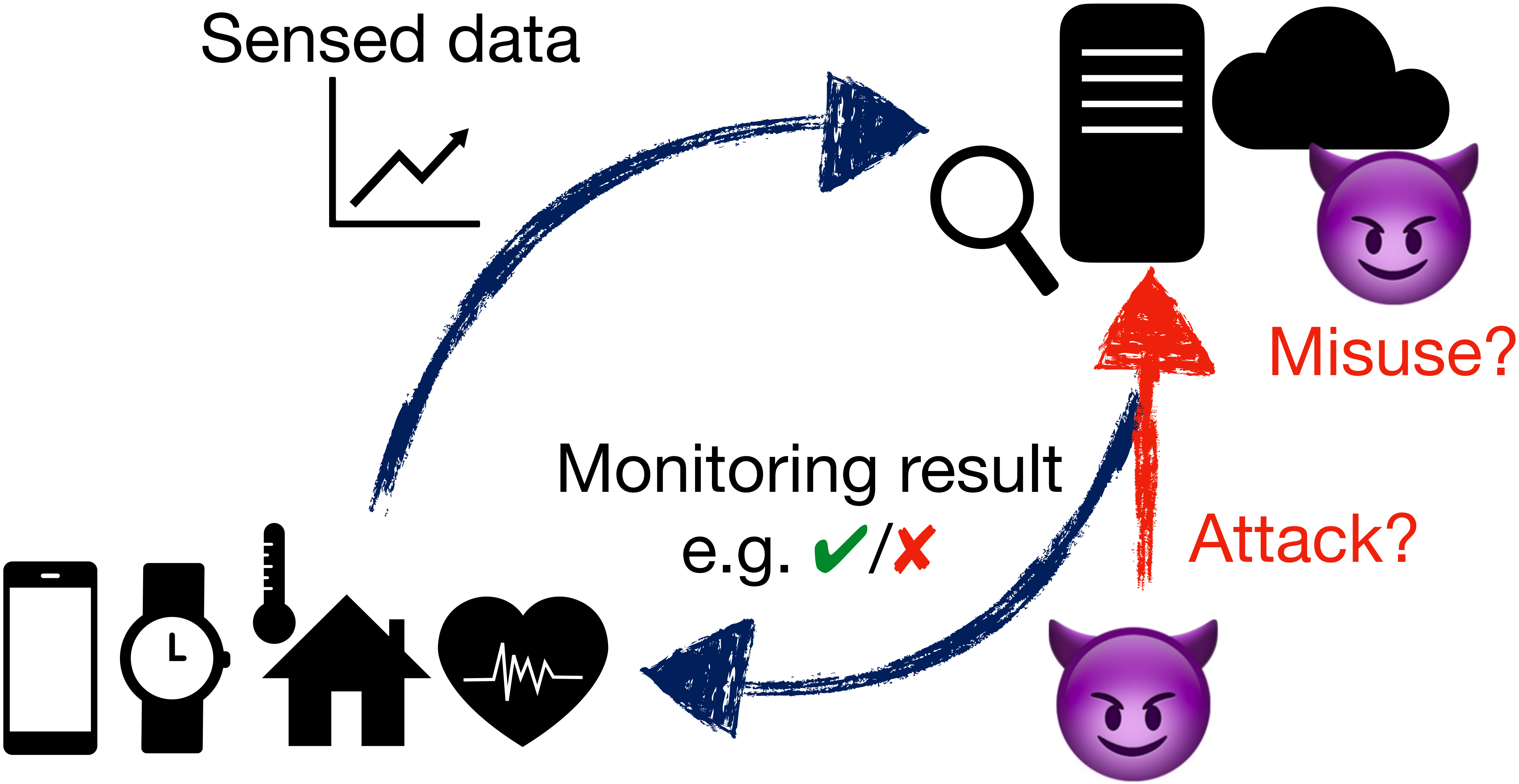
Monitoring with IoT is useful



Monitoring with IoT is useful ...but privacy?

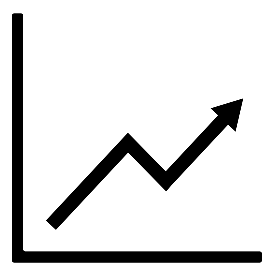


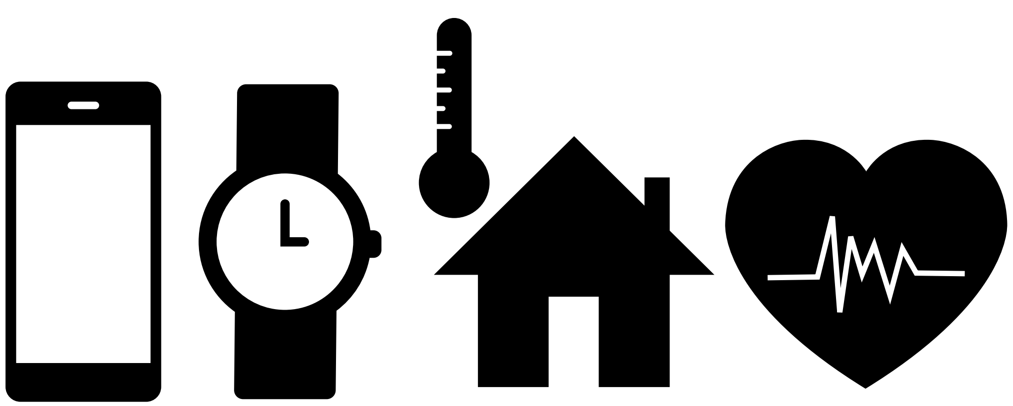
Monitoring with IoT is useful ...but privacy?



Monitoring with IoT is useful
...but privacy?

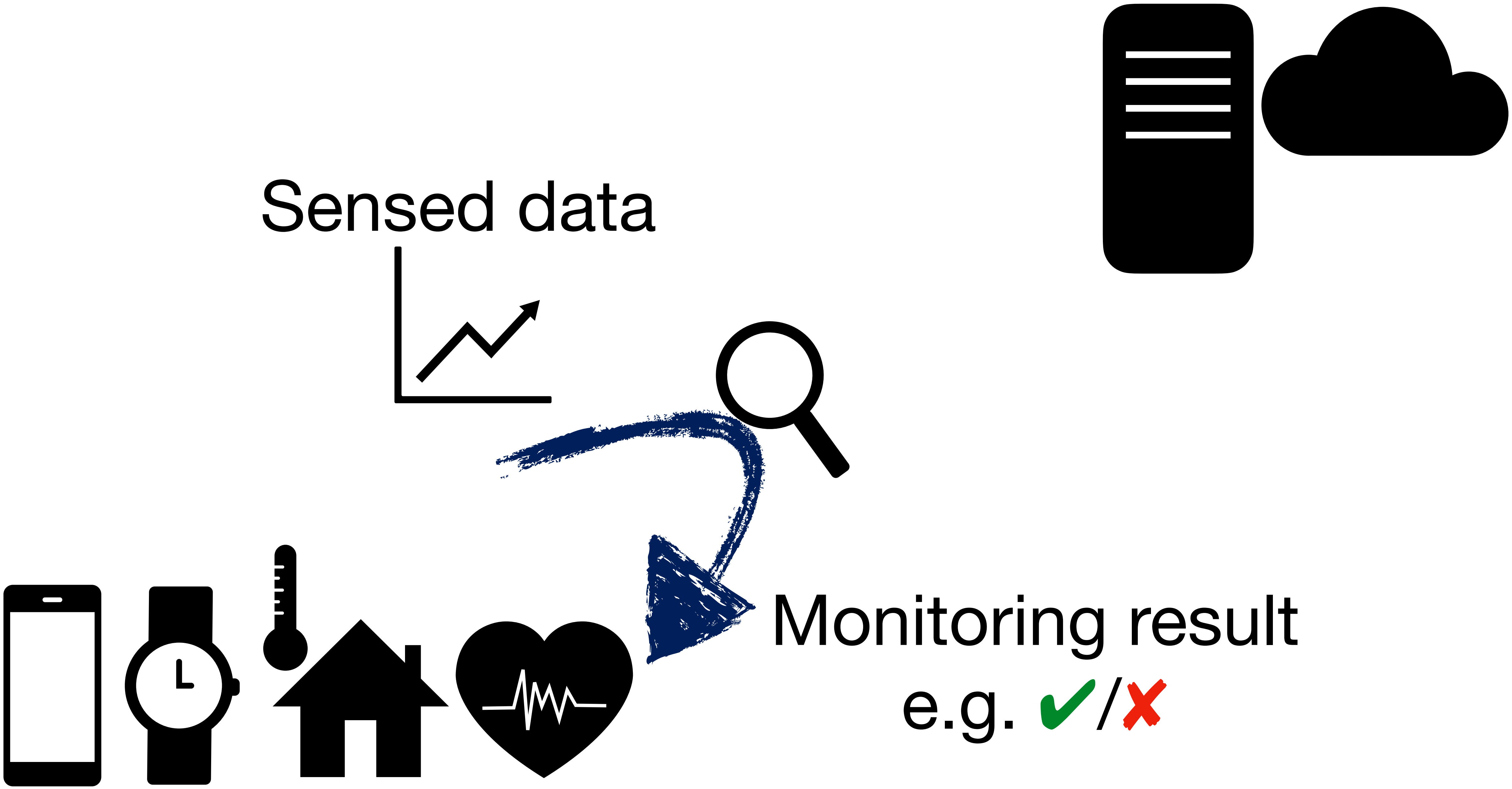
Sensed data



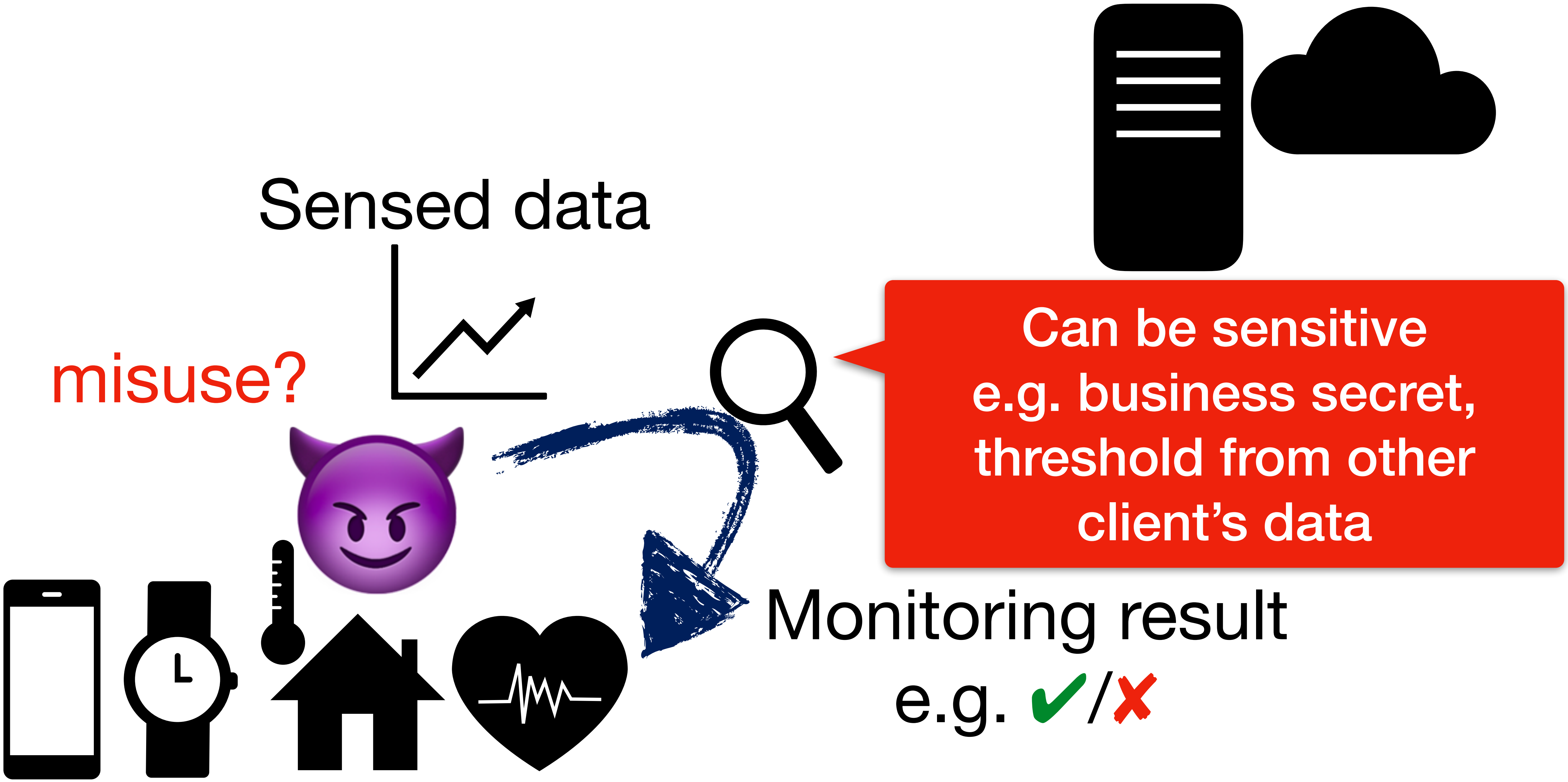


Monitoring result
e.g. ✔/✘

Monitoring with IoT is useful
...but privacy?



Monitoring with IoT is useful
...but privacy?



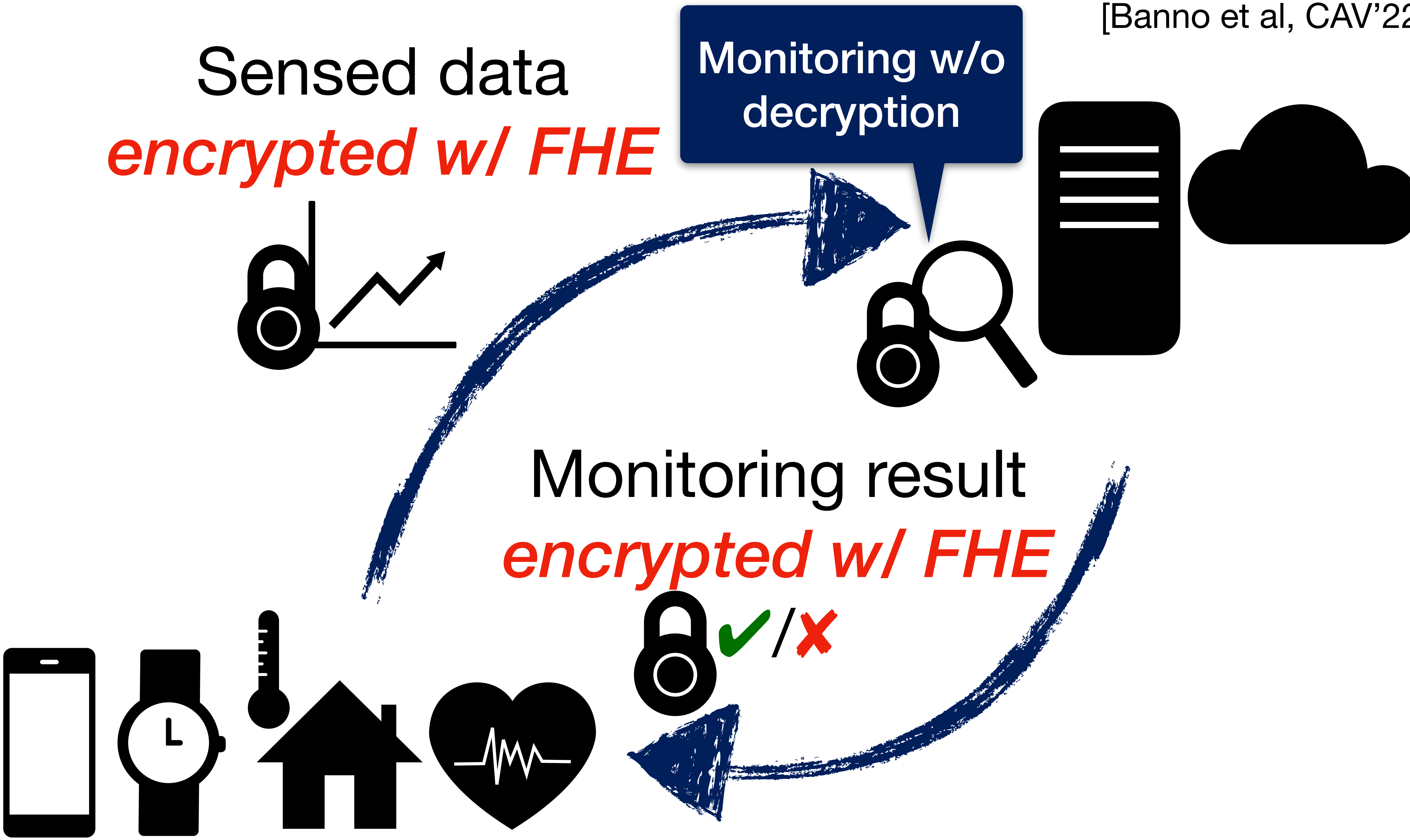
Q. Can server monitor without knowing the content?

Q. Can server monitor without knowing the content?

A. Yes, with Fully Homomorphic Encryption

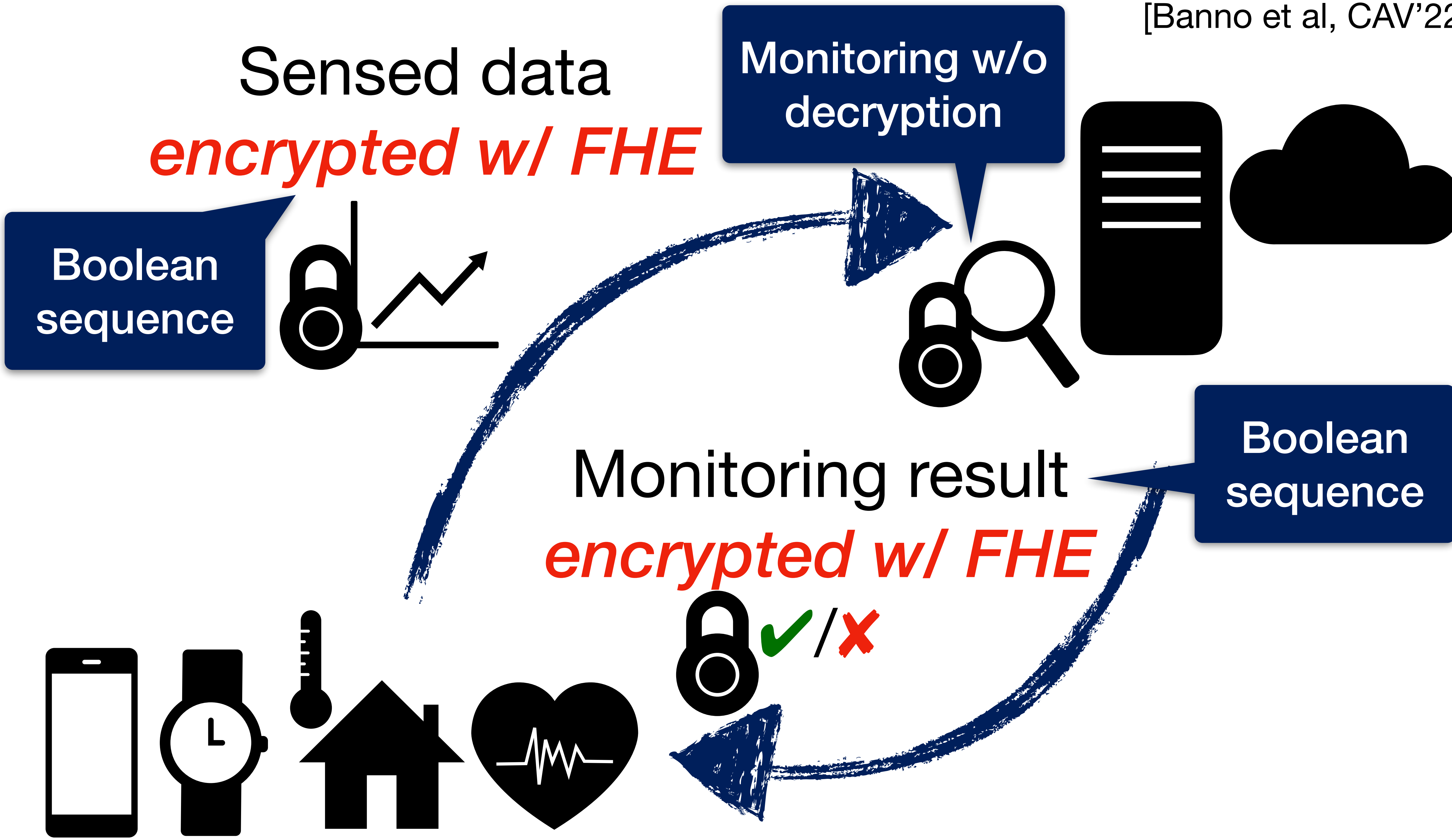
Oblivious Online LTL Monitoring

[Banno et al, CAV'22]



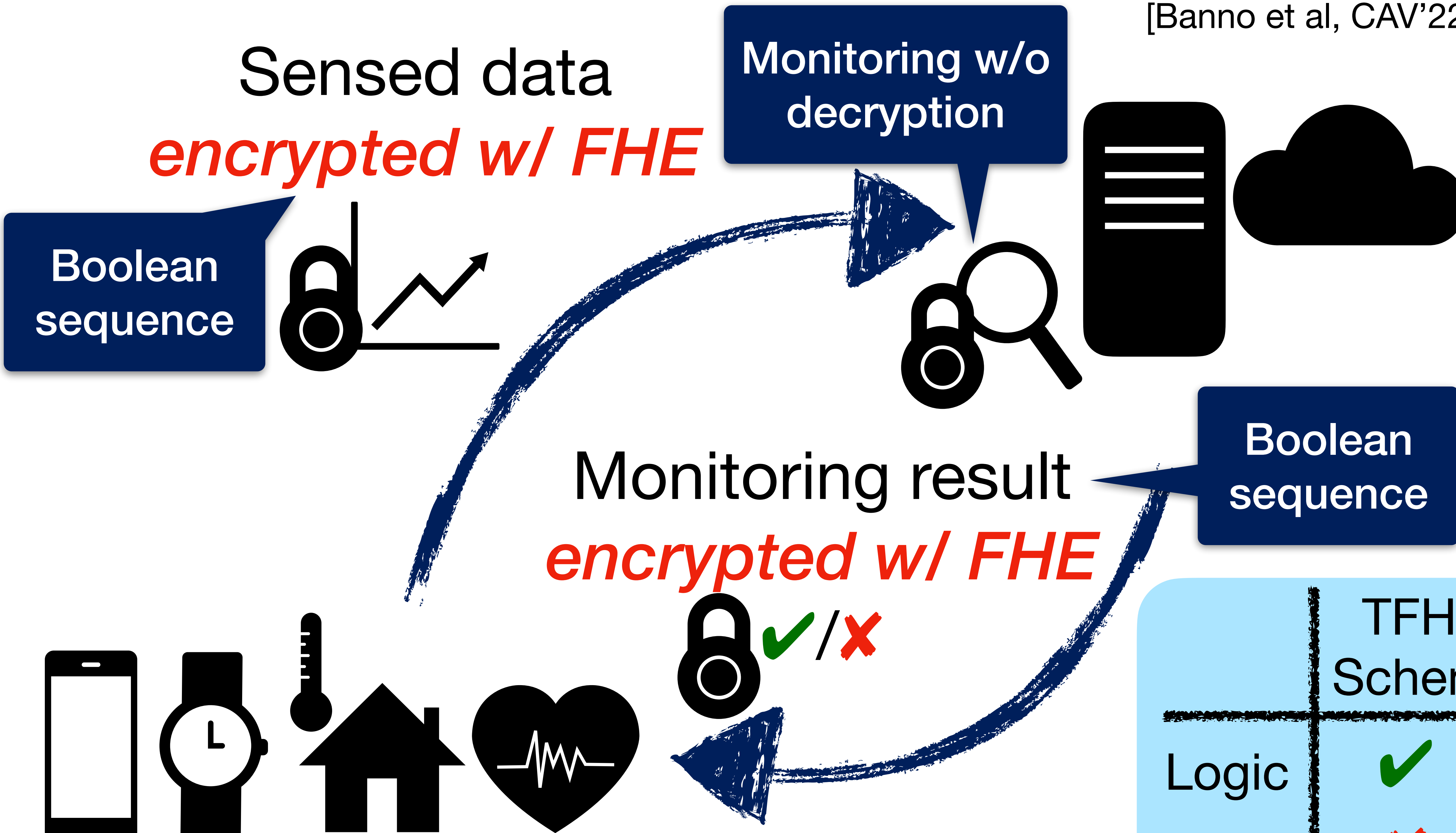
Oblivious Online LTL Monitoring

[Banno et al, CAV'22]



Oblivious Online LTL Monitoring

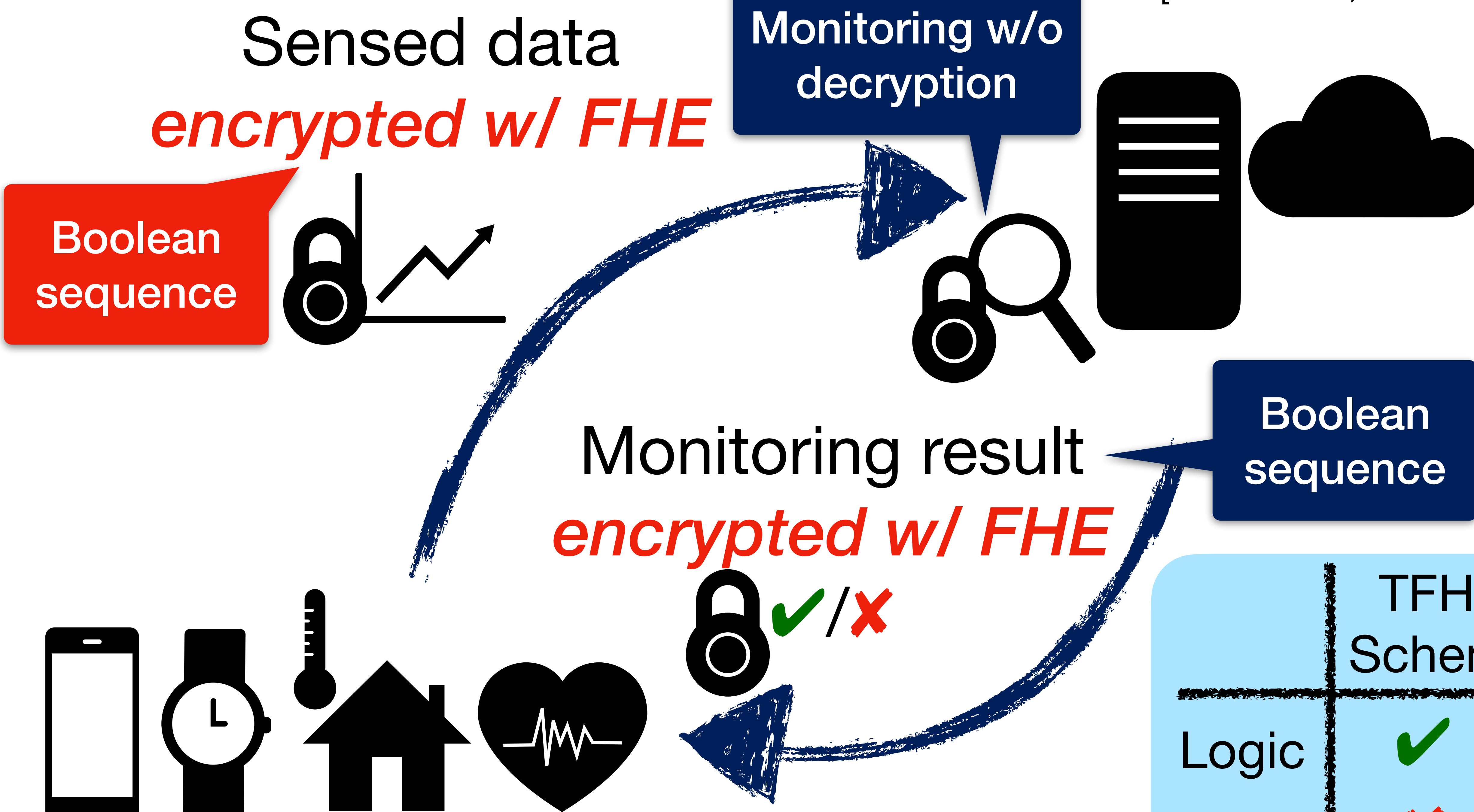
[Banno et al, CAV'22]



	TFHE Scheme
Logic	✔
Arith.	✘

Oblivious Online LTL Monitoring

[Banno et al, CAV'22]



	TFHE Scheme
Logic	✔
Arith.	✘

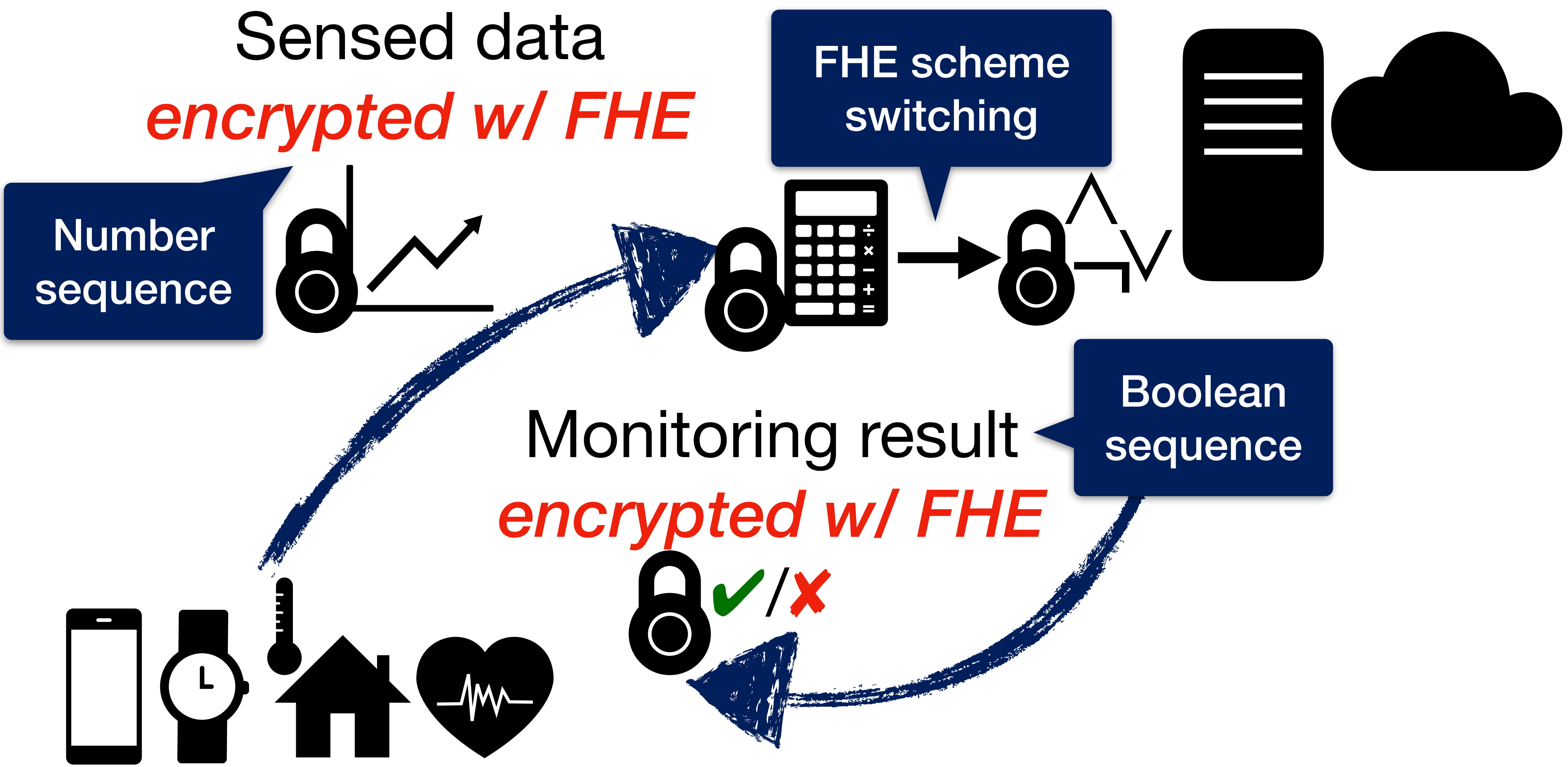
Q. Can we handle temporal + arith?

Q. Can we handle temporal + arith?

A. Yes, by bridging FHE schemes

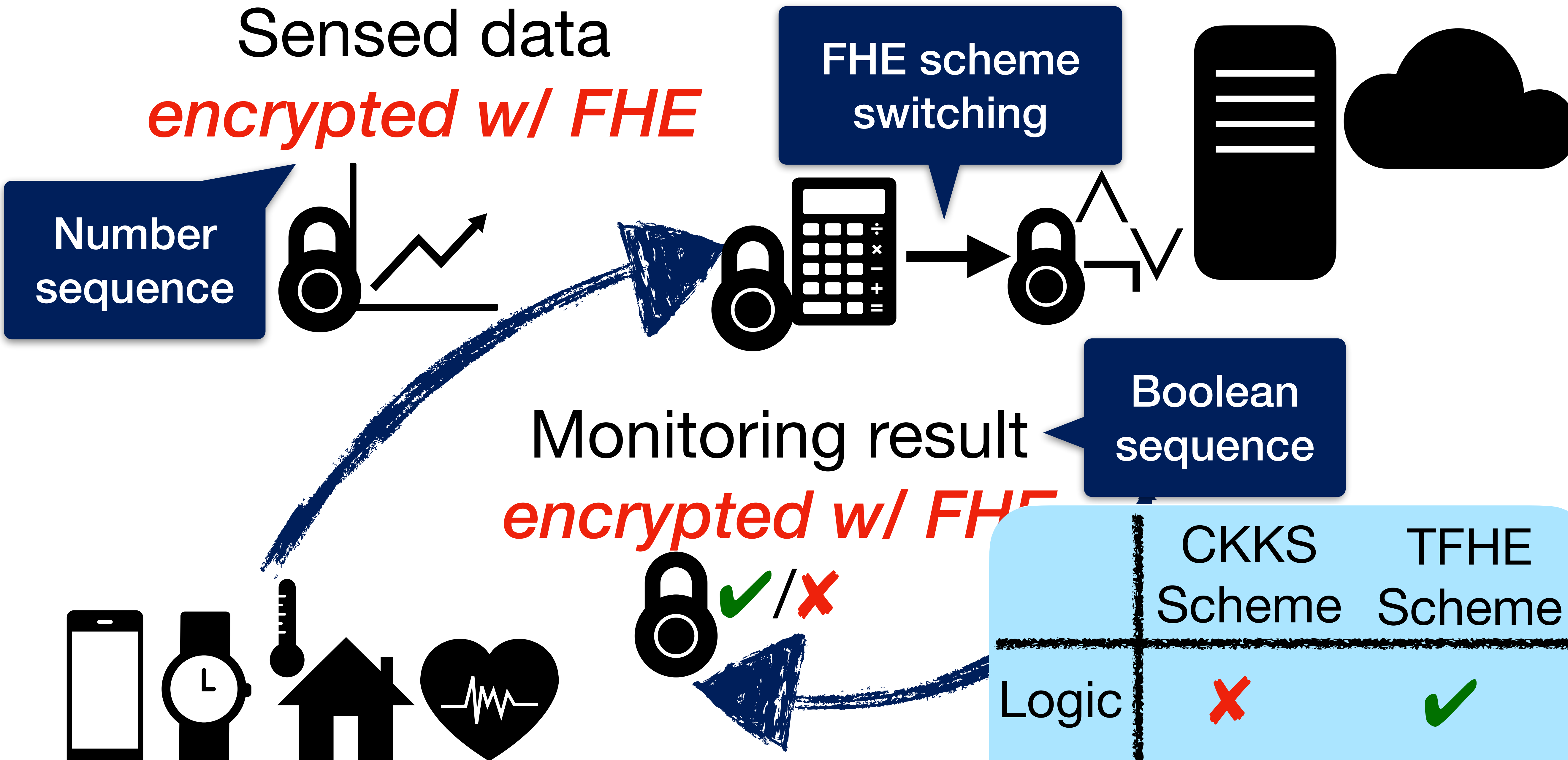
Oblivious Online STL Monitoring

[Contribution]



Oblivious Online STL Monitoring

[Contribution]



	CKKS Scheme	TFHE Scheme
Logic	✘	✔
Arith.	✔	✘
⋮		

Contributions

- Online oblivious discrete-time STL monitoring protocol
  - Combining CKKS and TFHE schemes
  - Note: discrete-time but with (linear) arith. predicates
- Optimization of scheme switching for STL monitoring
  - The largest technical contribution
- Implementation + experiments
  → Fast enough for blood glucose monitoring
  Works for RSS monitoring, too

Outline

- Review: Oblivious LTL monitoring with FHE
- Oblivious discrete-time STL monitoring with FHE
  - Overview of the workflow
  - Optimization of scheme switching
- Experiments

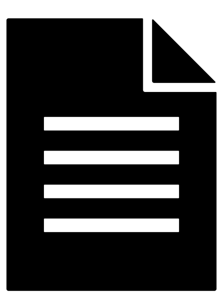
Fully Homomorphic Encryption

Normal Encryption (e.g. RSA) | **Fully Homomorphic Encryption**

Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)

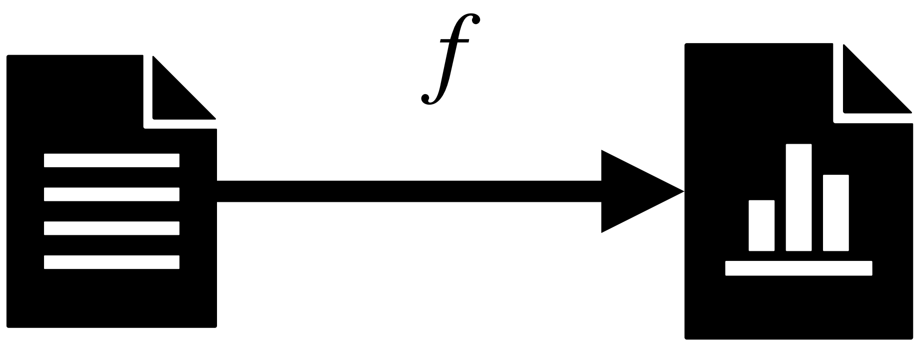
Fully Homomorphic Encryption



Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)

Fully Homomorphic Encryption

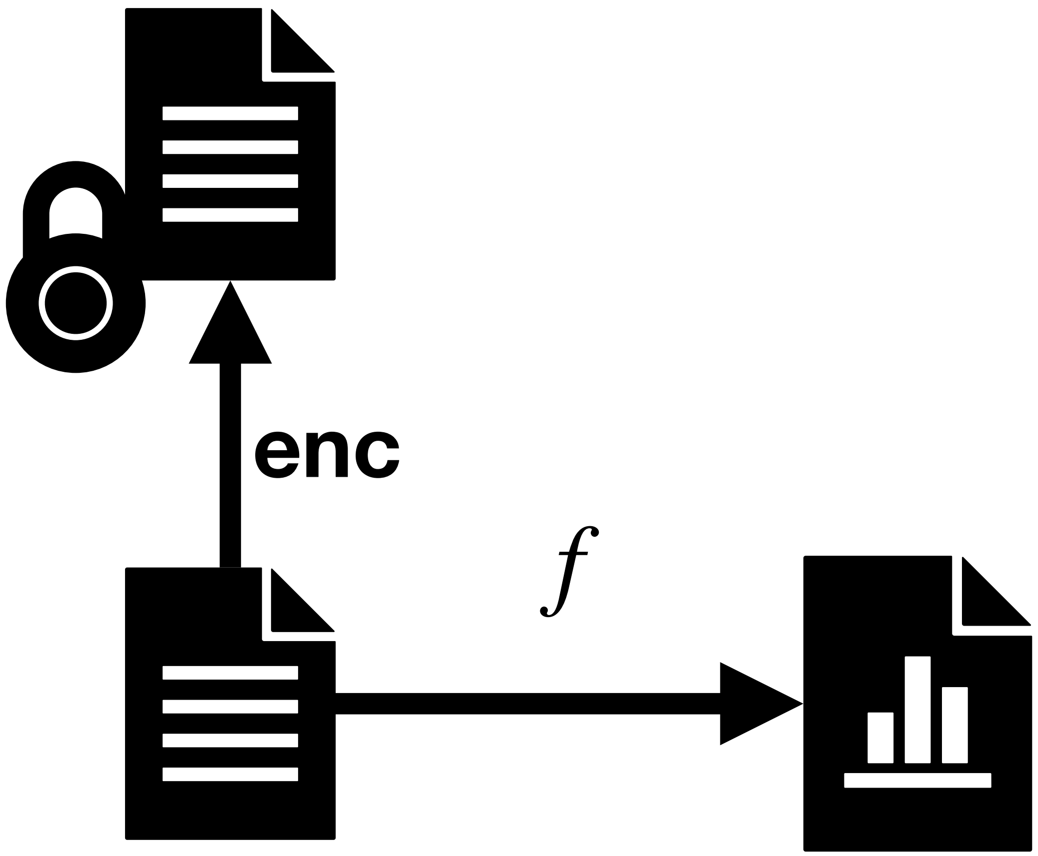


M. Waga (Kyoto U.)

Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)

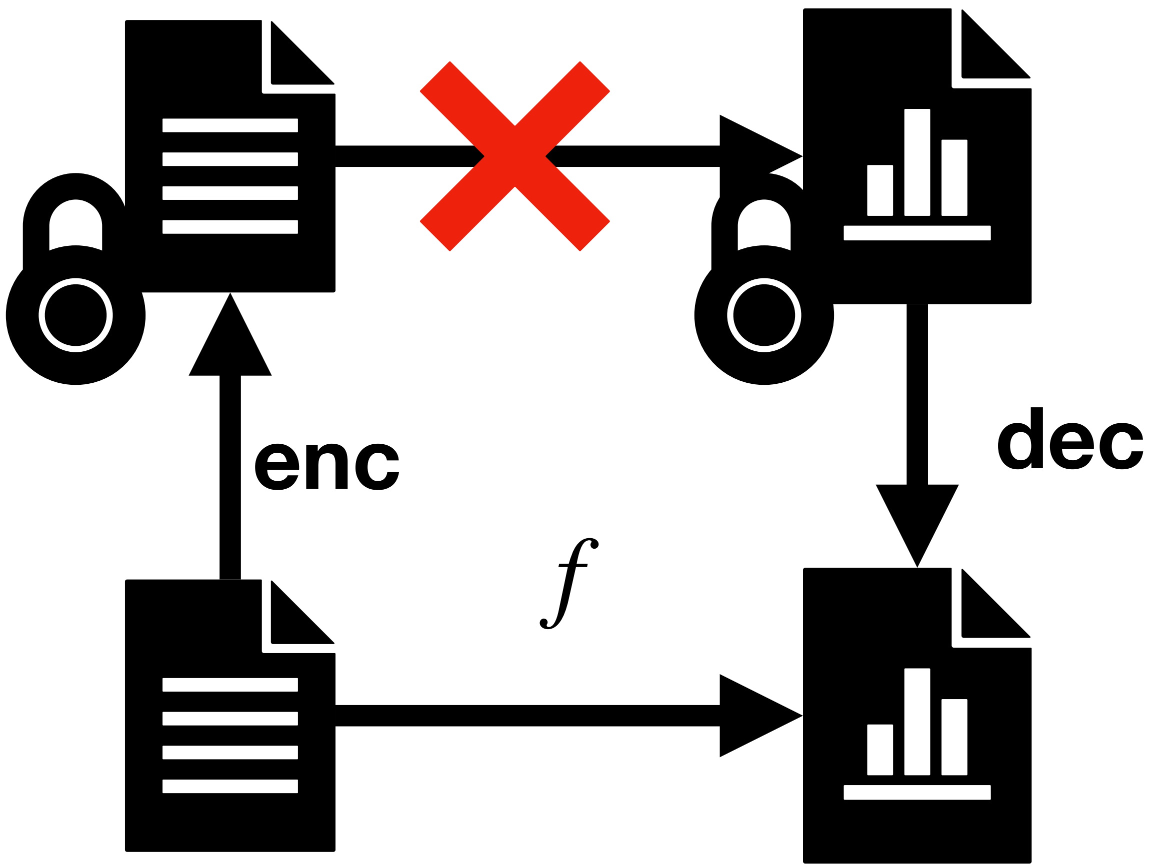
Fully Homomorphic Encryption



Fully Homomorphic Encryption

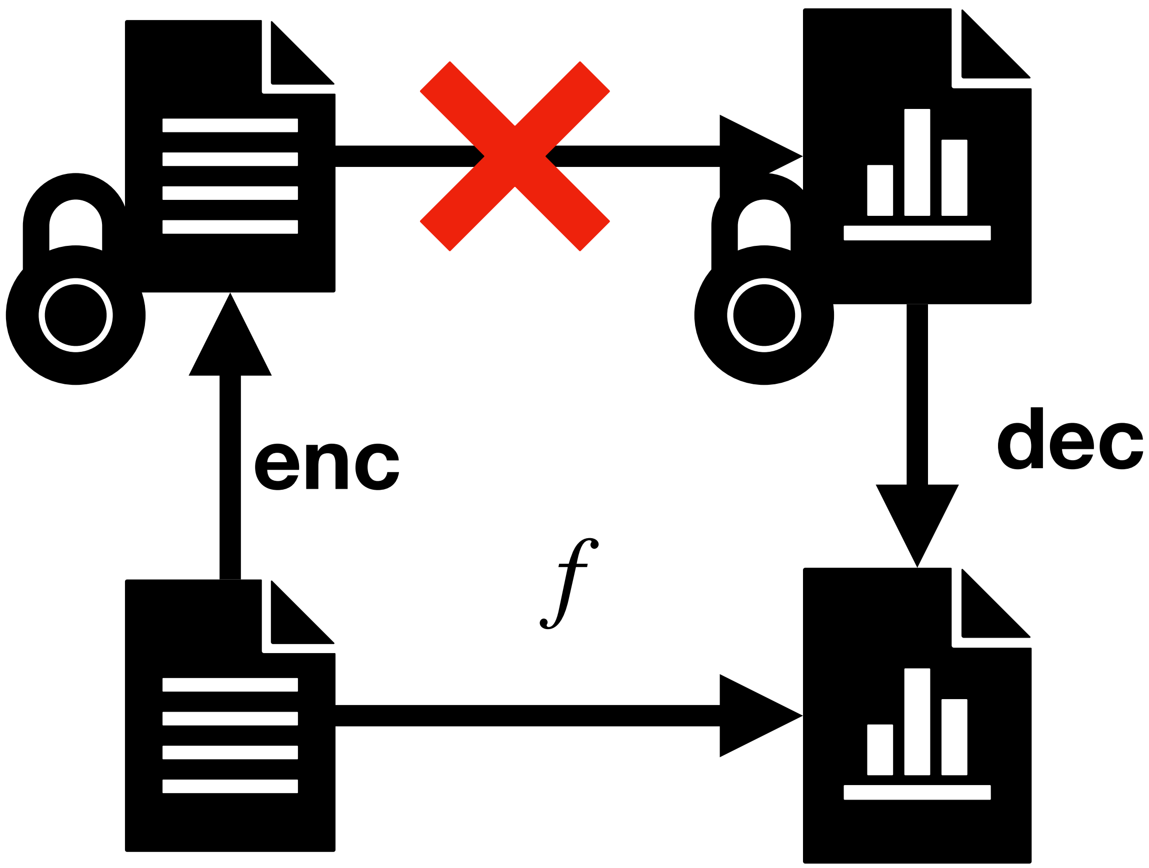
Normal Encryption (e.g. RSA)

Fully Homomorphic Encryption

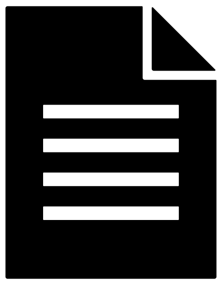


Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)

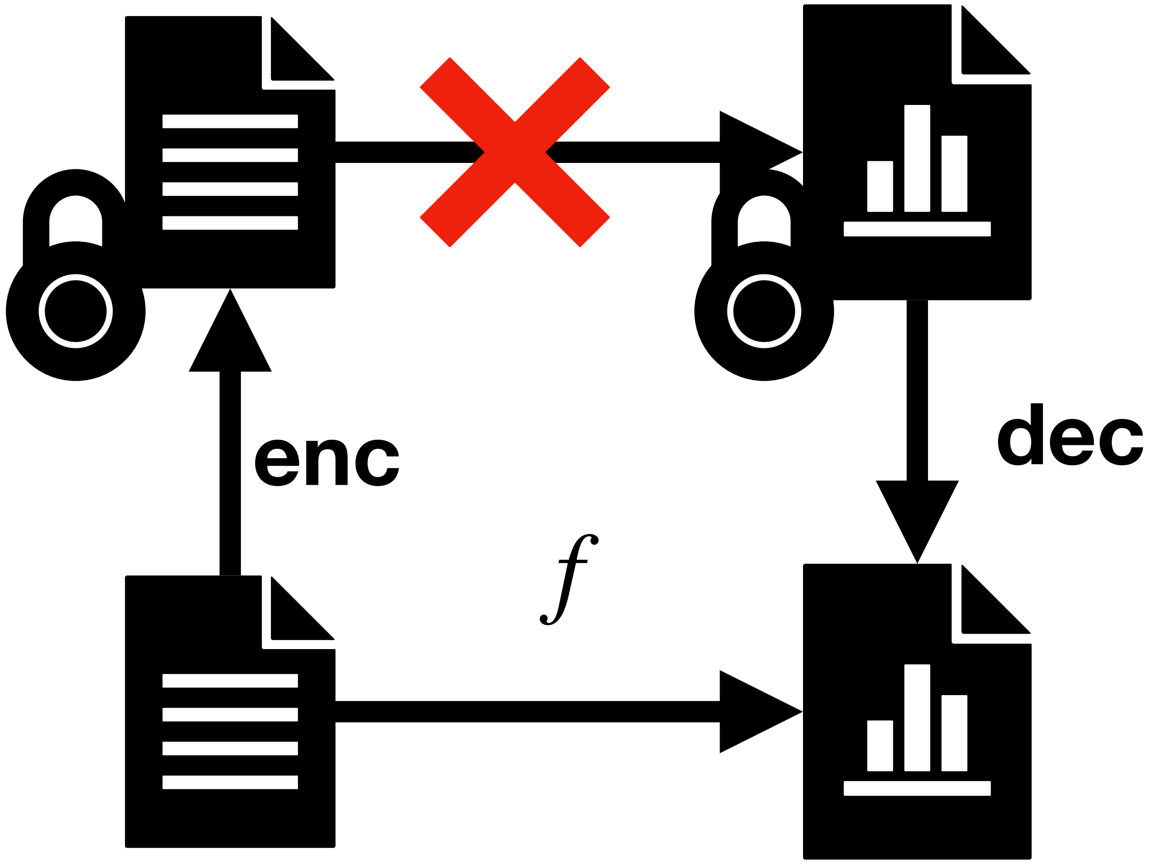


Fully Homomorphic Encryption

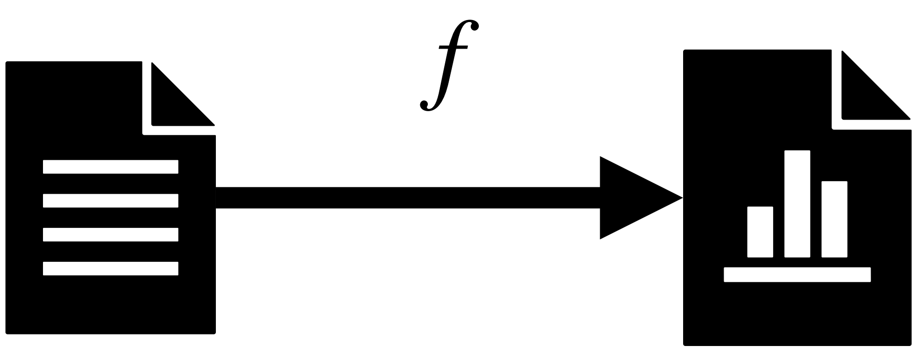


Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)

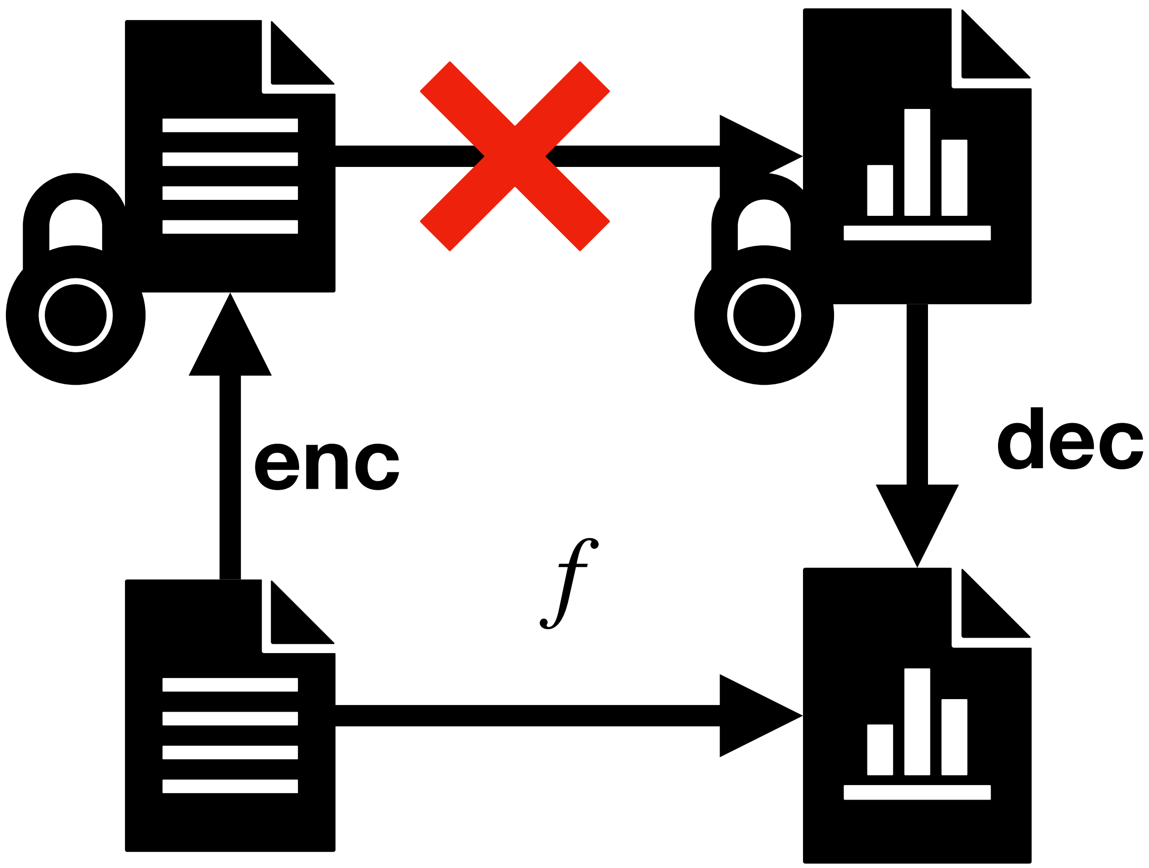


Fully Homomorphic Encryption

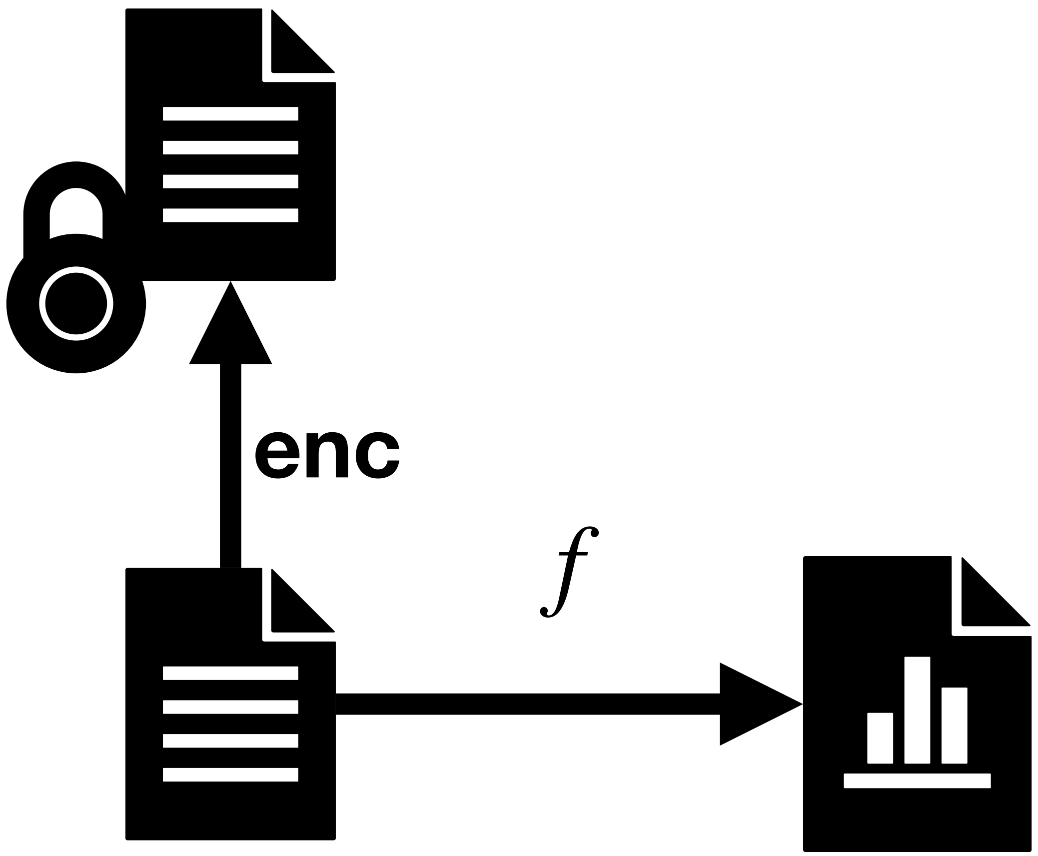


Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)

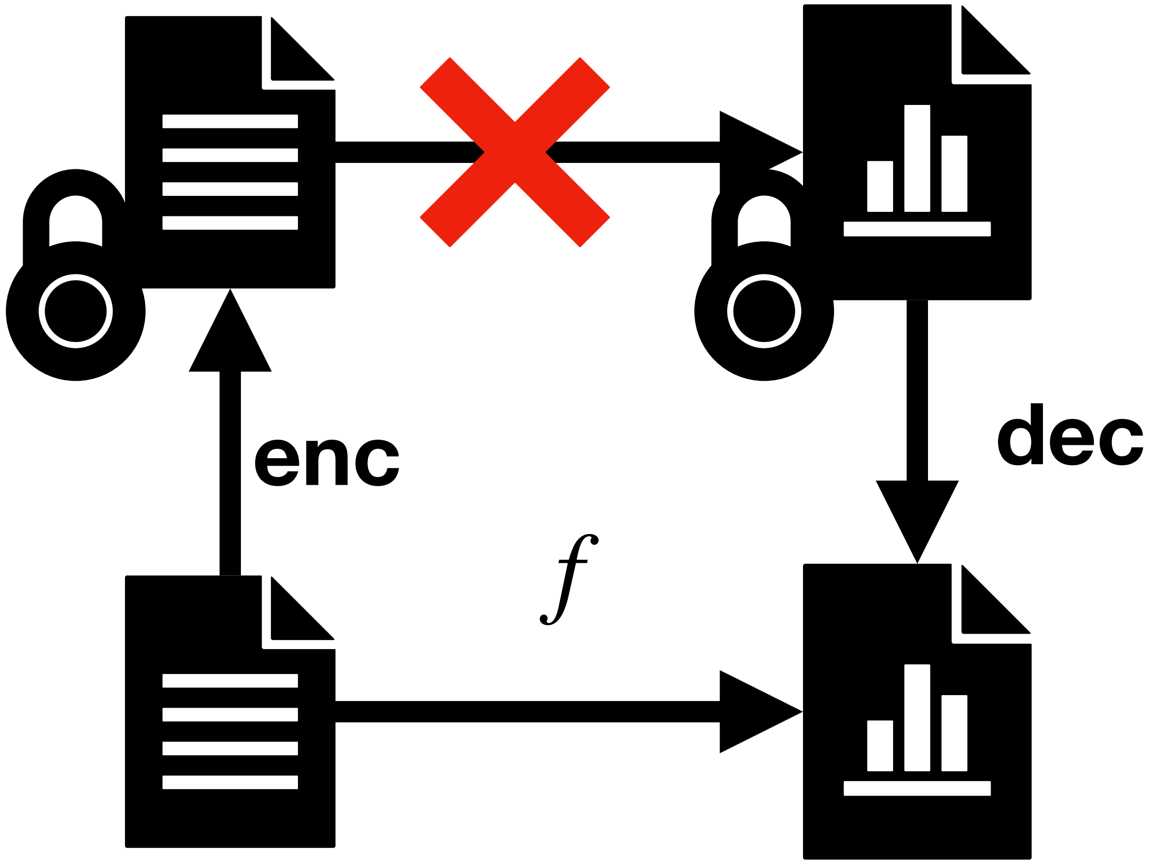


Fully Homomorphic Encryption

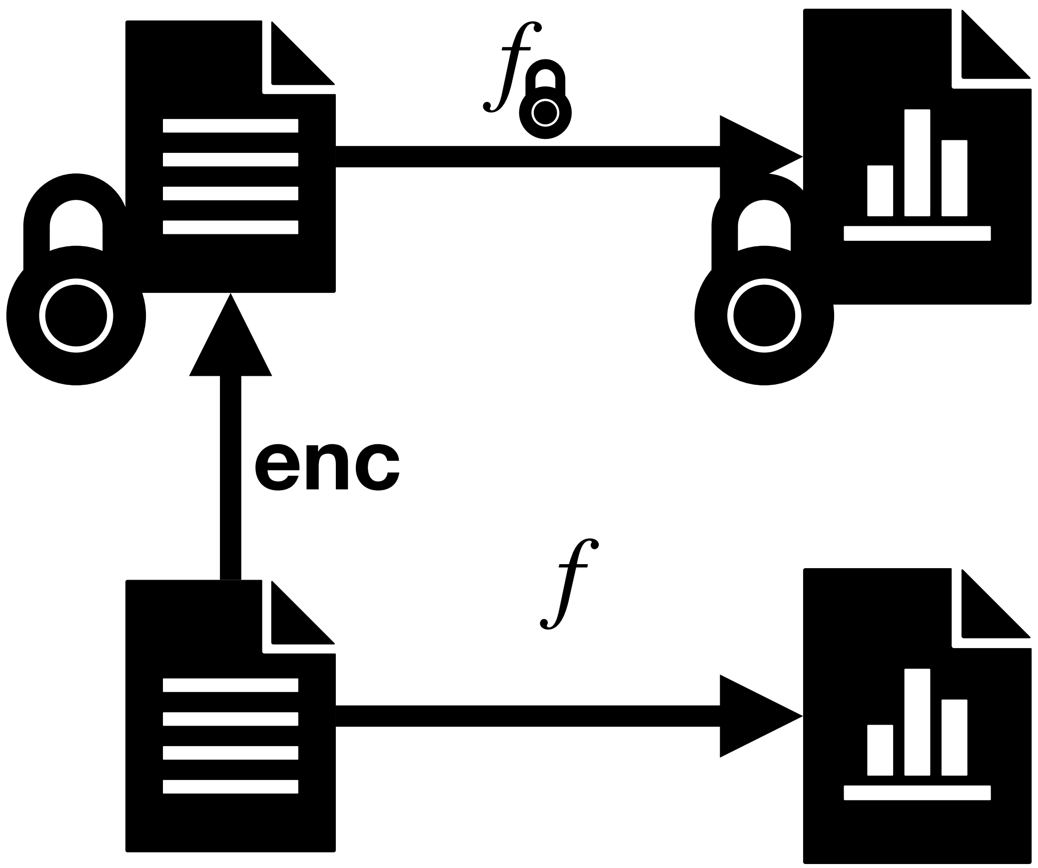


Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)

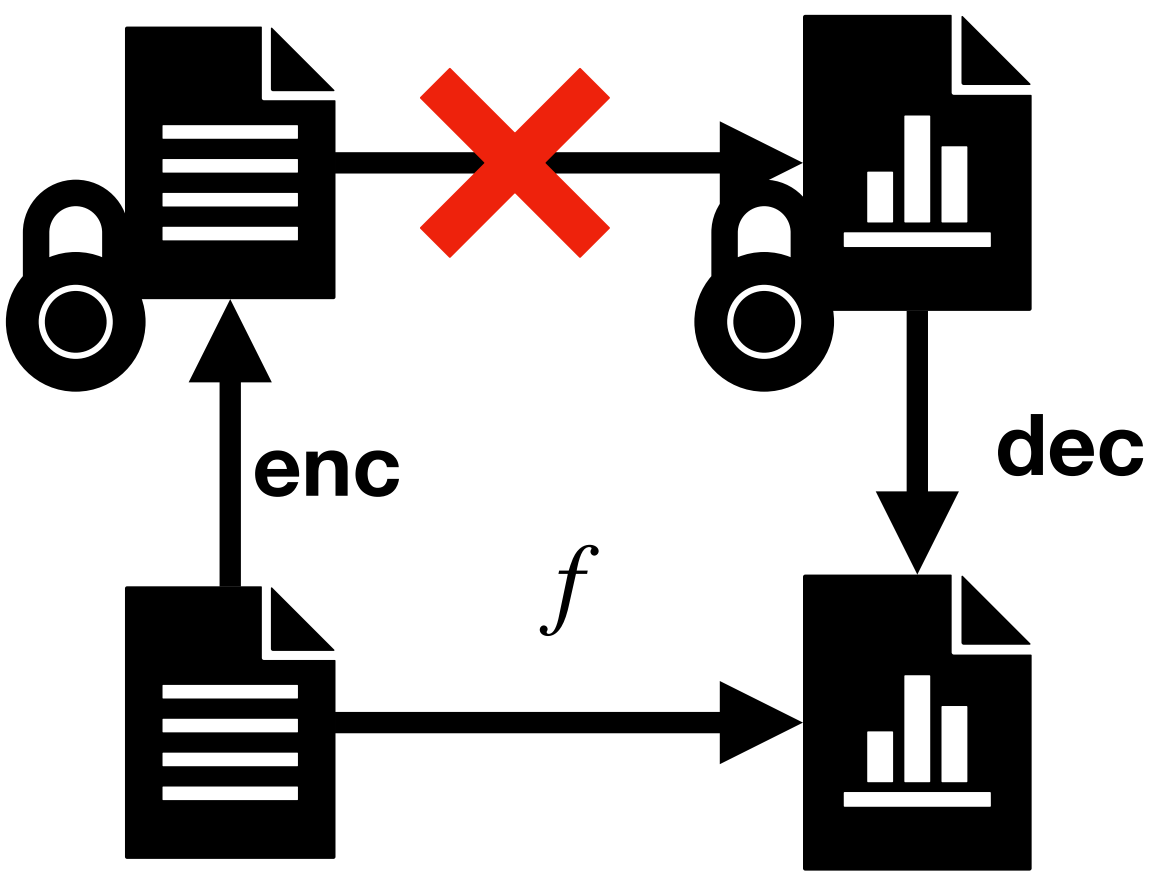


Fully Homomorphic Encryption

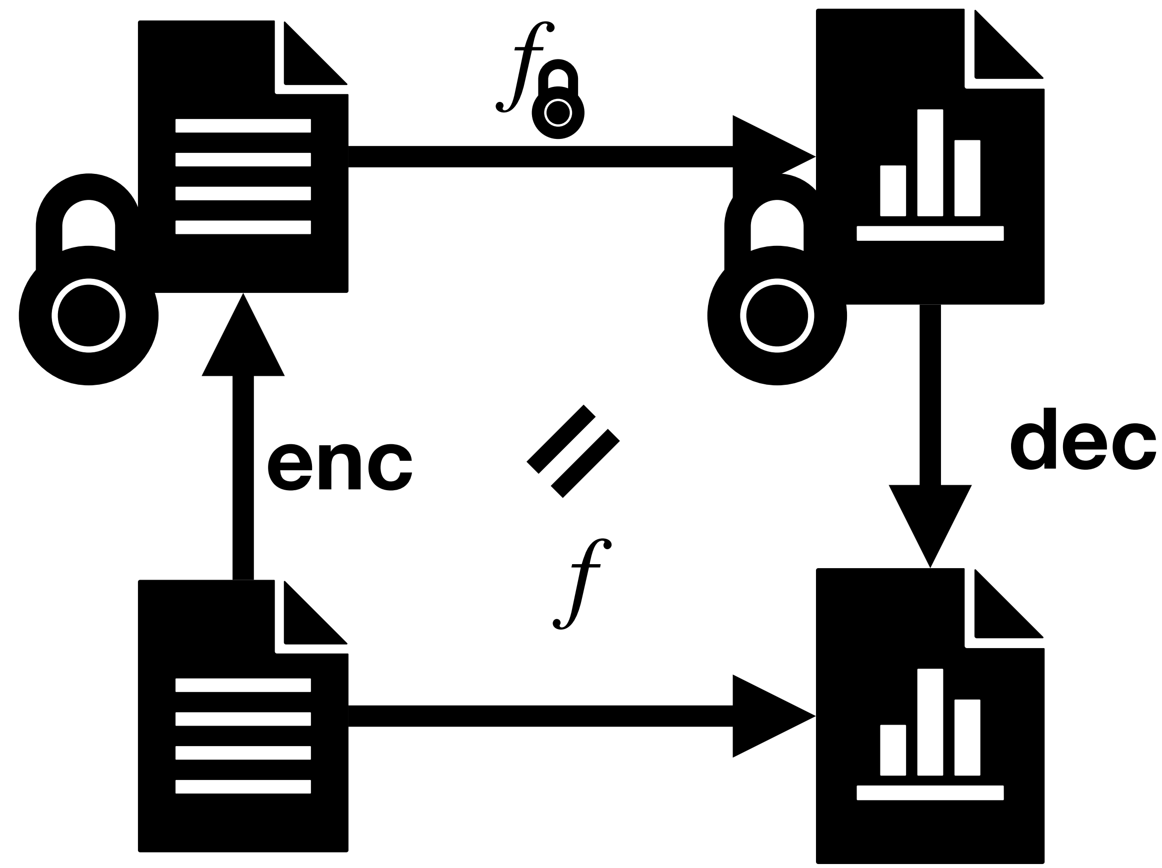


Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)

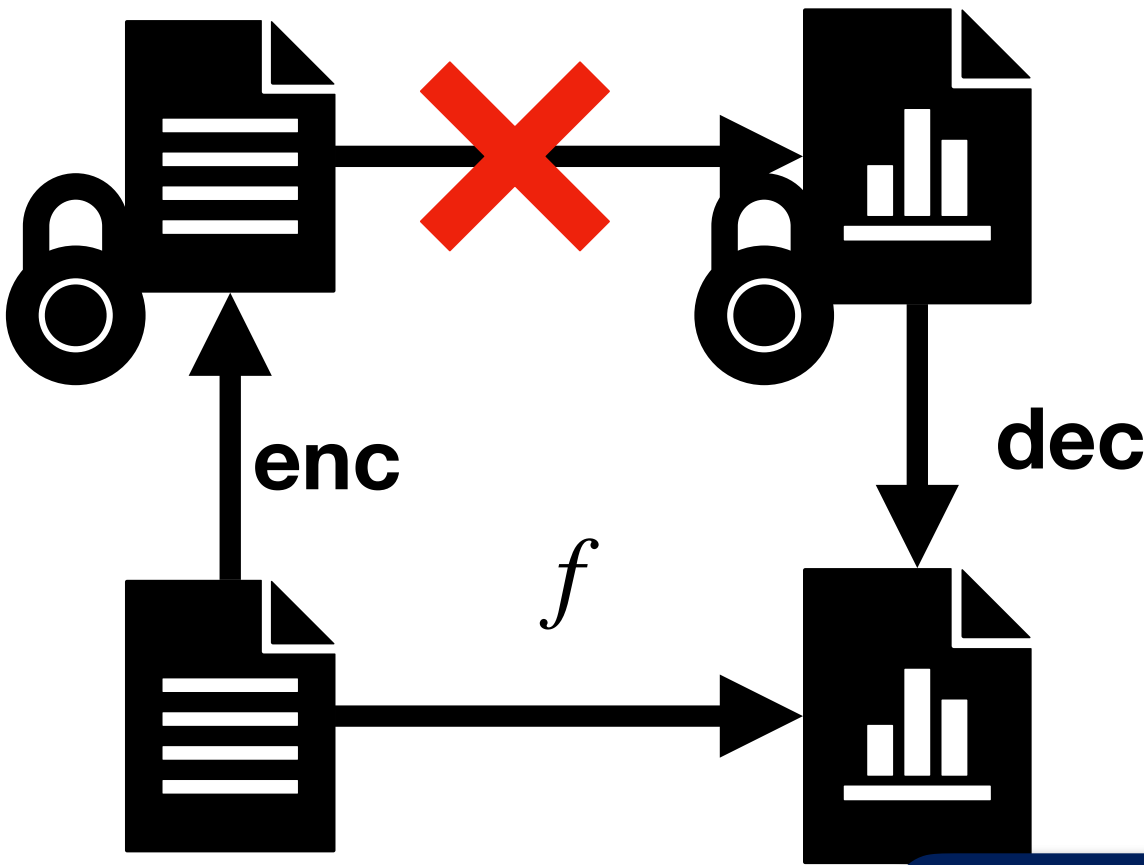


Fully Homomorphic Encryption

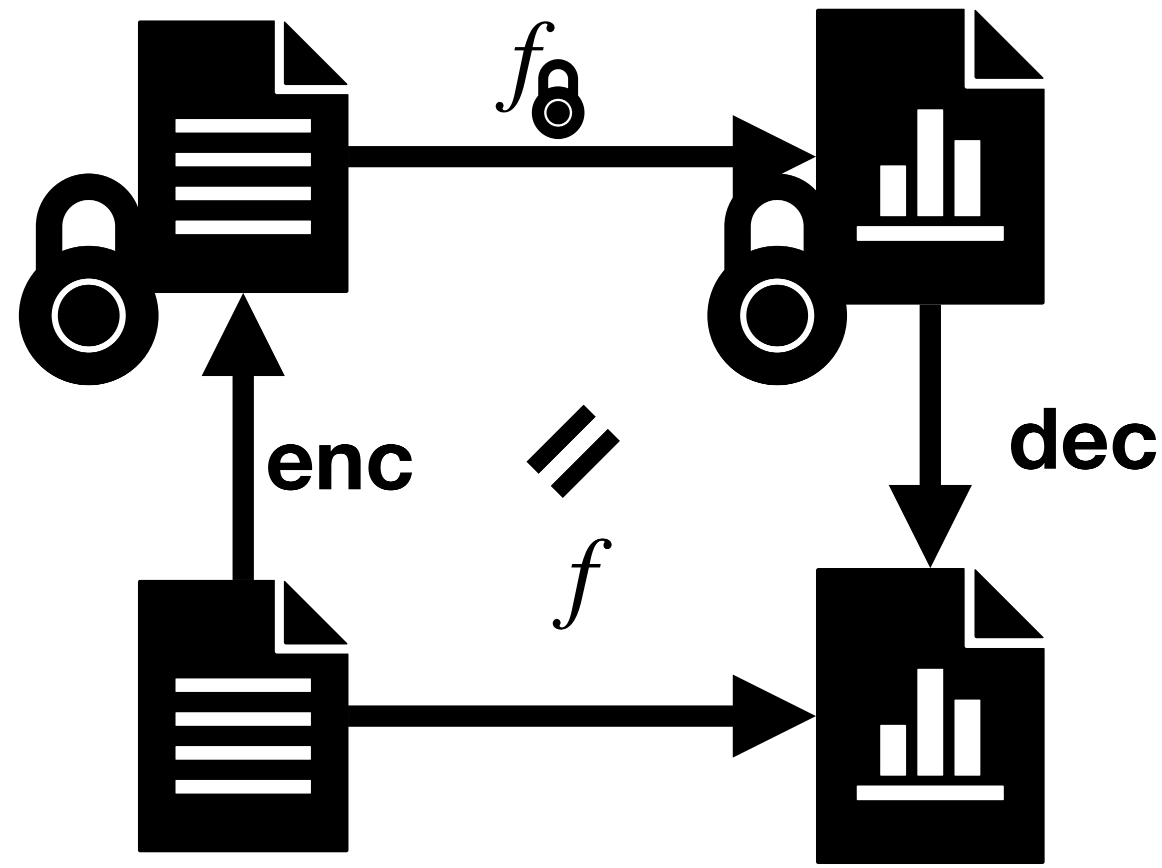


Fully Homomorphic Encryption

Normal Encryption (e.g. RSA)



Fully Homomorphic Encryption



Remark: Naive computation is usually very slow
→ dedicated algorithm is necessary
e.g. VM with FHE is a few Hz

Ciphertexts in FHE: (R)LWE

Noisy ciphertexts/operations for security

- (Low-level) Ciphertext: (masked) message + “noise”
- (Noisy) Operations
  - Noisy encryption
  - Noisy addition
  - Noisy multiplication
  - …
- Special Operation: Bootstrapping

Ciphertexts in FHE: (R)LWE

Noisy ciphertexts/operations for security

- (Low-level) Ciphertext: (masked) message + “noise”
- (Noisy) Operations
  - Noisy encryption
  - Noisy addition
  - Noisy multiplication
  - …

Increases noise
→ Eventually breaks message

- Special Operation: Bootstrapping

Ciphertexts in FHE: (R)LWE

Noisy ciphertexts/operations for security

- (Low-level) Ciphertext: (masked) message + "noise"
- (Noisy) Operations
  - Noisy encryption
  - Noisy addition
  - Noisy multiplication
  - …

Increases noise
→ Eventually breaks message

- Special Operation: Bootstrapping

Reduces noise

Ciphertexts in FHE: (R)LWE

Noisy ciphertexts/operations for security

- (Low-level) Ciphertext: (masked) message + “noise”

> Result is always approx.
> → We need high-level “scheme”

- (Noisy) Operations
  - Noisy encryption
  - Noisy addition
  - Noisy multiplication
  - …

> Increases noise
> → Eventually breaks message

- Special Operation: Bootstrapping

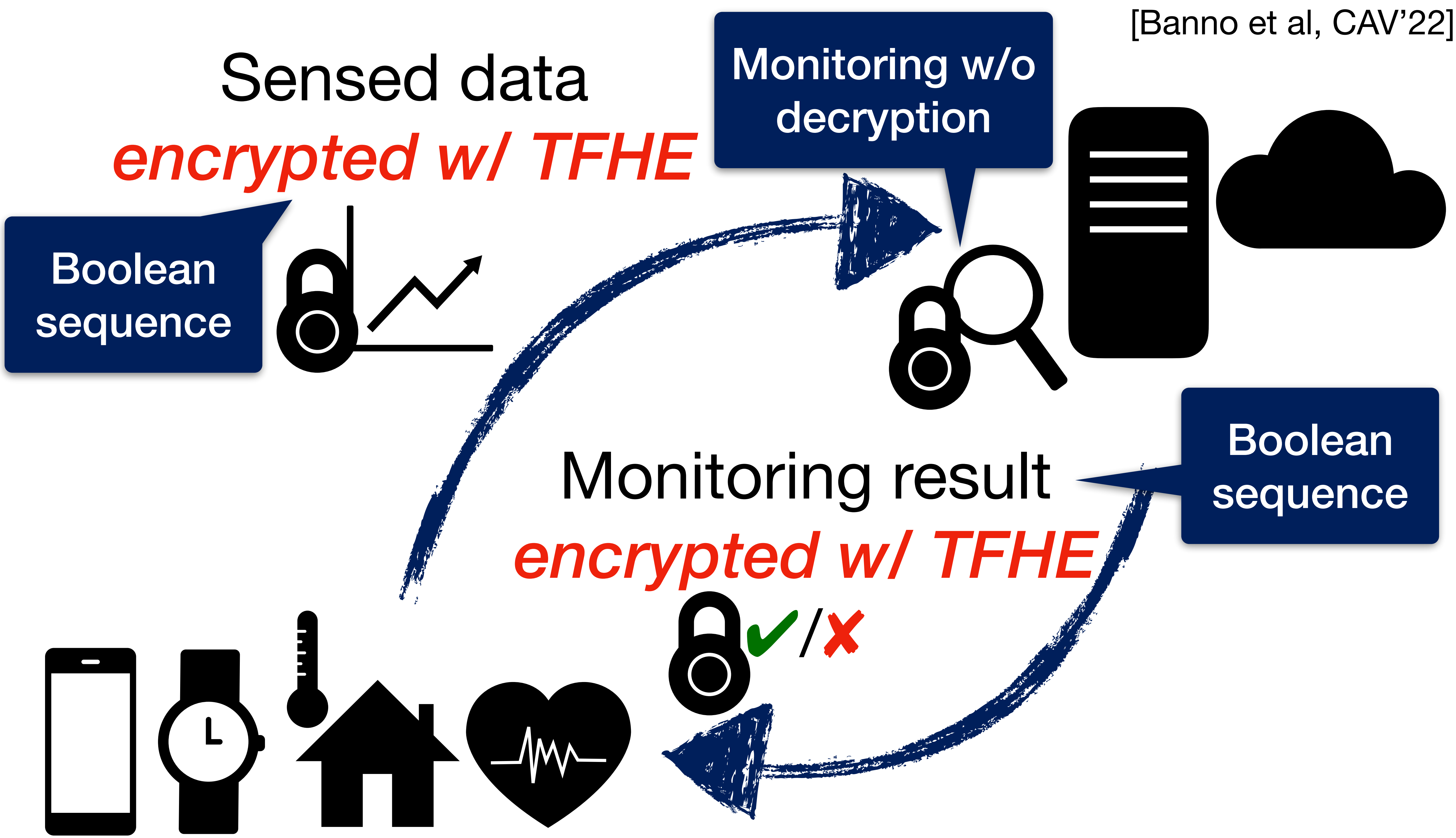
> Reduces noise

CKKS vs. TFHE Schemes

	CKKS [Cheon et al., ASIACRYPT'17]	TFHE [Chillotti et al., J. Crypto '20]
Typical Usage	Approx. Numbers w/ (linear) Arith. Op. e.g. +, -, *	(Exact) Booleans with logical operations e.g. and, or, not, nand
Bootstrapping for noise reduction	Very Slow e.g. > 20 sec. Can be ≈ 1.5 min.	(Relatively) Fast e.g. < 500 ms.

Values are taken from Al Badawi, Ahmad, and Yuriy Polyakov. "Demystifying bootstrapping in fully homomorphic encryption." Cryptology ePrint Archive (2023).

Oblivious Online LTL Monitoring



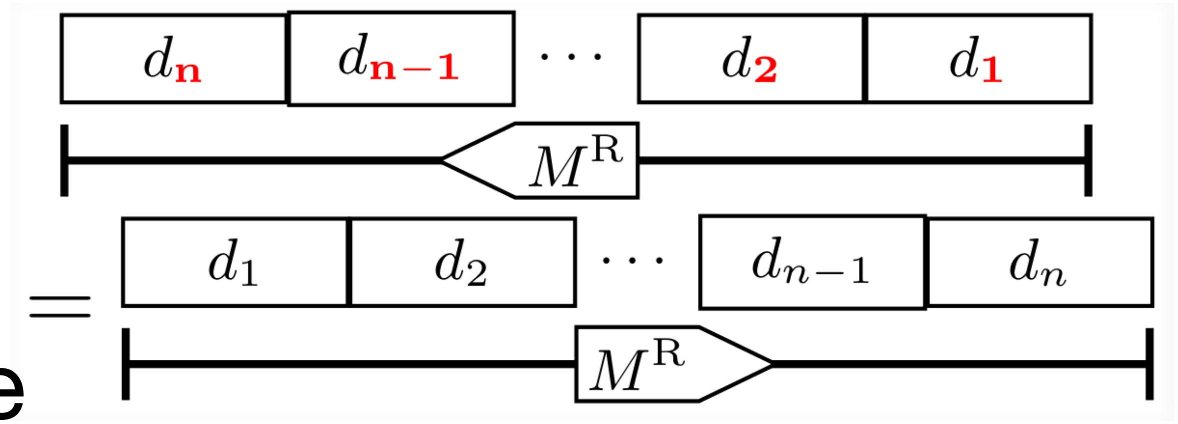
Oblivious Online DFA Execution

[Banno et al, CAV'22]

Two algorithms for FHE-based DFA execution

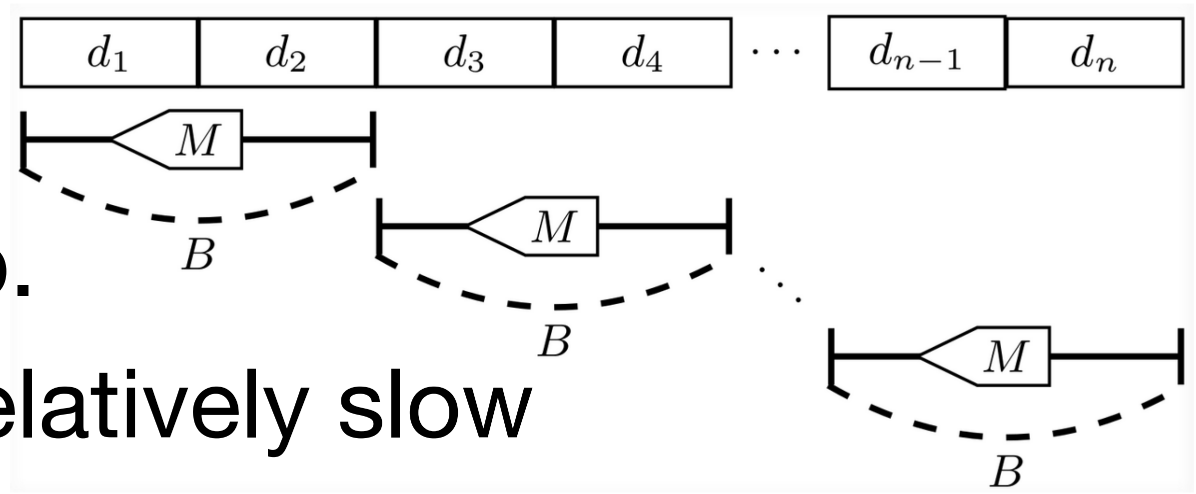
Reverse

- Reverse the given DFA
- Reversed DFA can be huge



Block

- "Blocked" backward comp.
- Jumping to next block is relatively slow



Outline

- Review: Oblivious LTL monitoring with FHE
- Oblivious discrete-time STL monitoring with FHE
  - Overview of the workflow
  - Optimization of scheme switching
- Experiments

Discrete-time STL: LTL + arith.

$$\varphi, \varphi' ::= \mu > c \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathscr{X}\varphi' \mid \varphi \mathscr{U}_{[i,j)} \varphi'$$

Discrete-time STL: LTL + arith.

$$\varphi, \varphi' ::= \mu > c \mid \neg\varphi \mid \varphi \lor \varphi \mid \mathscr{X}\varphi' \mid \varphi\,\mathscr{U}_{[i,j)}\varphi'$$

(Linear) arith. predicate

Same as LTL
→ Can be converted to a DFA

Discrete-time STL: LTL + arith.

$$\varphi, \varphi' ::= \mu > c \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathcal{X}\varphi' \mid \varphi \mathcal{U}_{[i,j)} \varphi'$$

(Linear) arith. predicate

Same as LTL
→ Can be converted to a DFA

# of arith. op. in μ is known (to the server)
→ Bootstrapping is unnec. for appropriate param.

Log length is unbounded
→ Need bootstrapping

Discrete-time STL: LTL + arith.

$$\varphi, \varphi' ::= \mu > c \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathcal{X}\varphi' \mid \varphi\,\mathcal{U}_{[i,j)}\varphi'$$

(Linear) arith. predicate

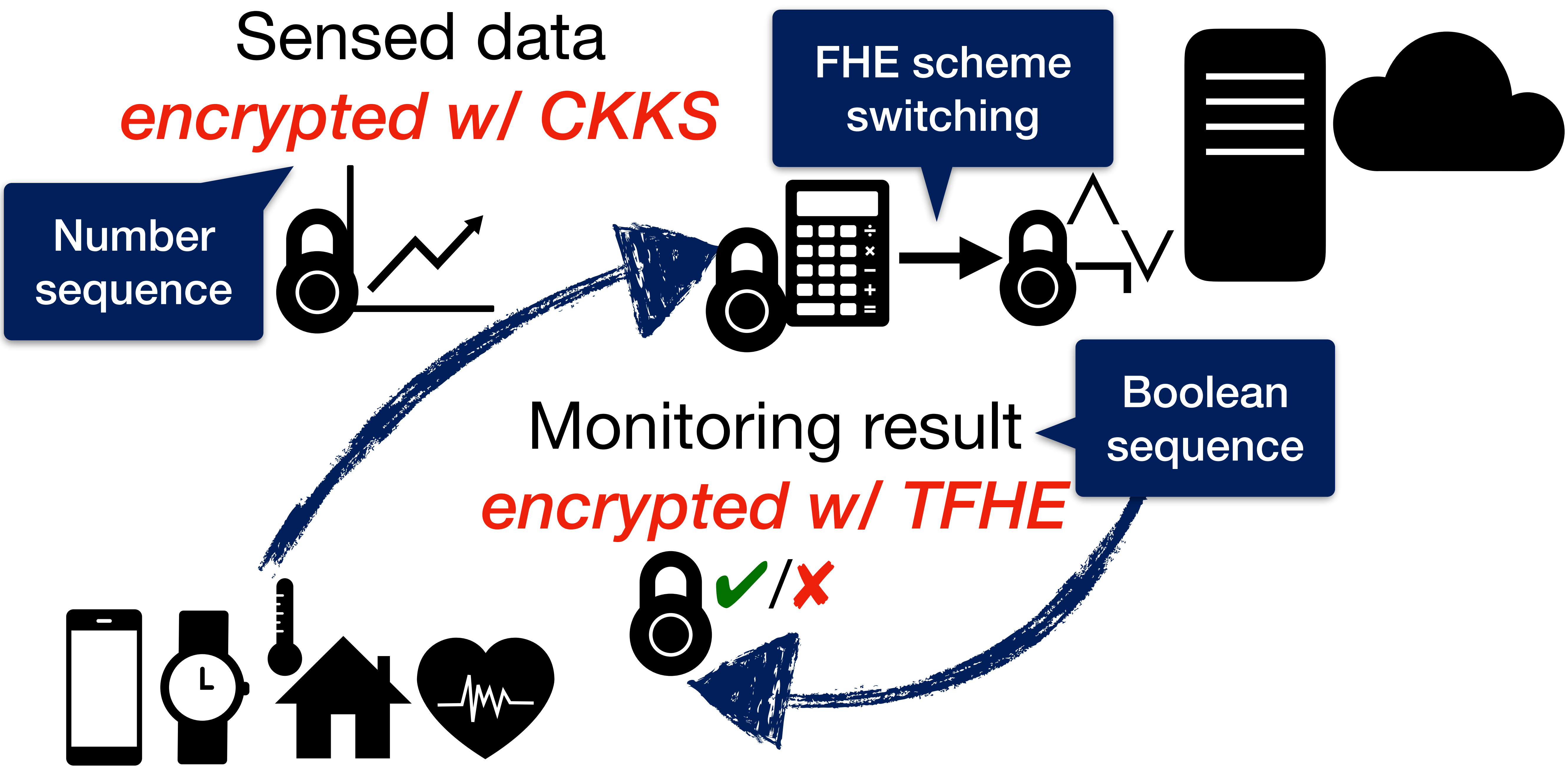
Same as LTL
→ Can be converted to a DFA

# of arith. op. in μ is known (to the server)
→ Bootstrapping is unnec. for appropriate param.

Log length is unbounded
→ Need bootstrapping

	CKKS [Cheon et al., ASIACRYPT'17]	TFHE [Chillotti et al., J. Crypto '20]
Typical Usage	Numbers	Booleans
Bootstrapping	Very Slow	(Relatively) Fast

Oblivious Online Monitoring of STL with arithmetic predicates [Contribution]

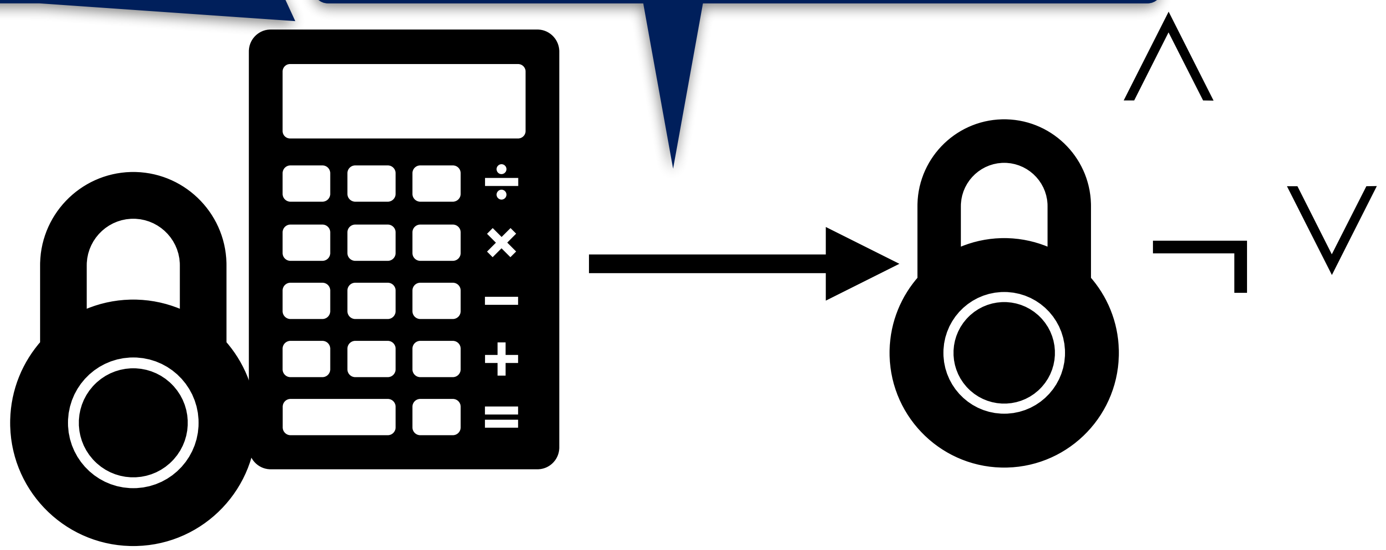


Oblivious Online Monitoring of STL with arithmetic predicates [Contribution]

1. Compute $\mu - c$ with CKKS

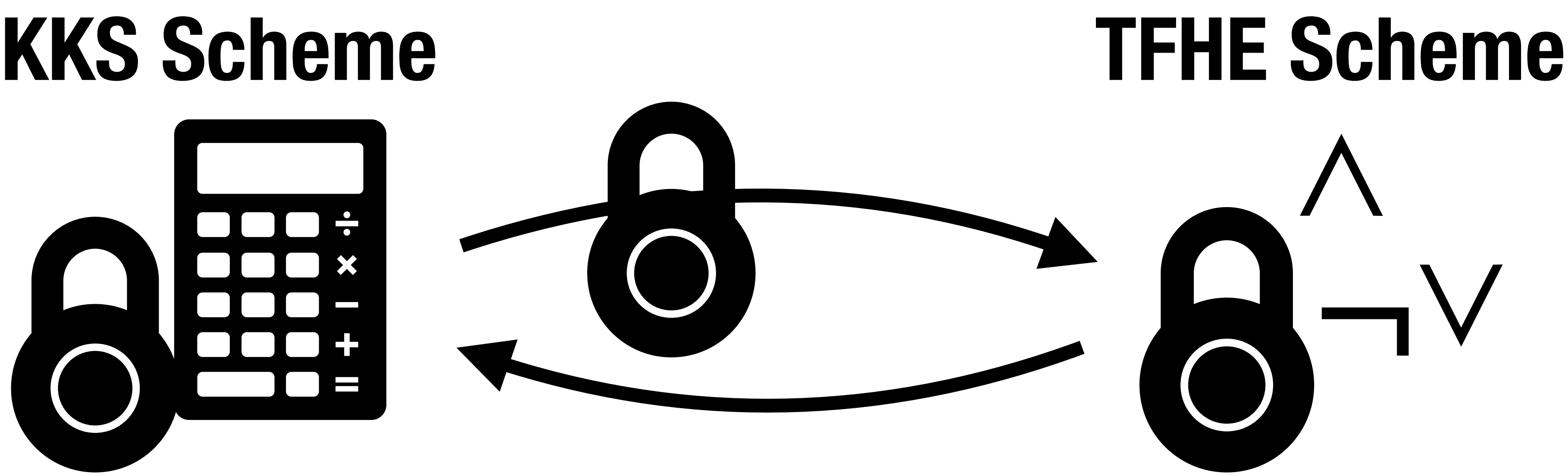
2. Obtain the truth value of $\mu > c$ as TFHE (sign of $\mu - c$)

3. Execute DFA with TFHE



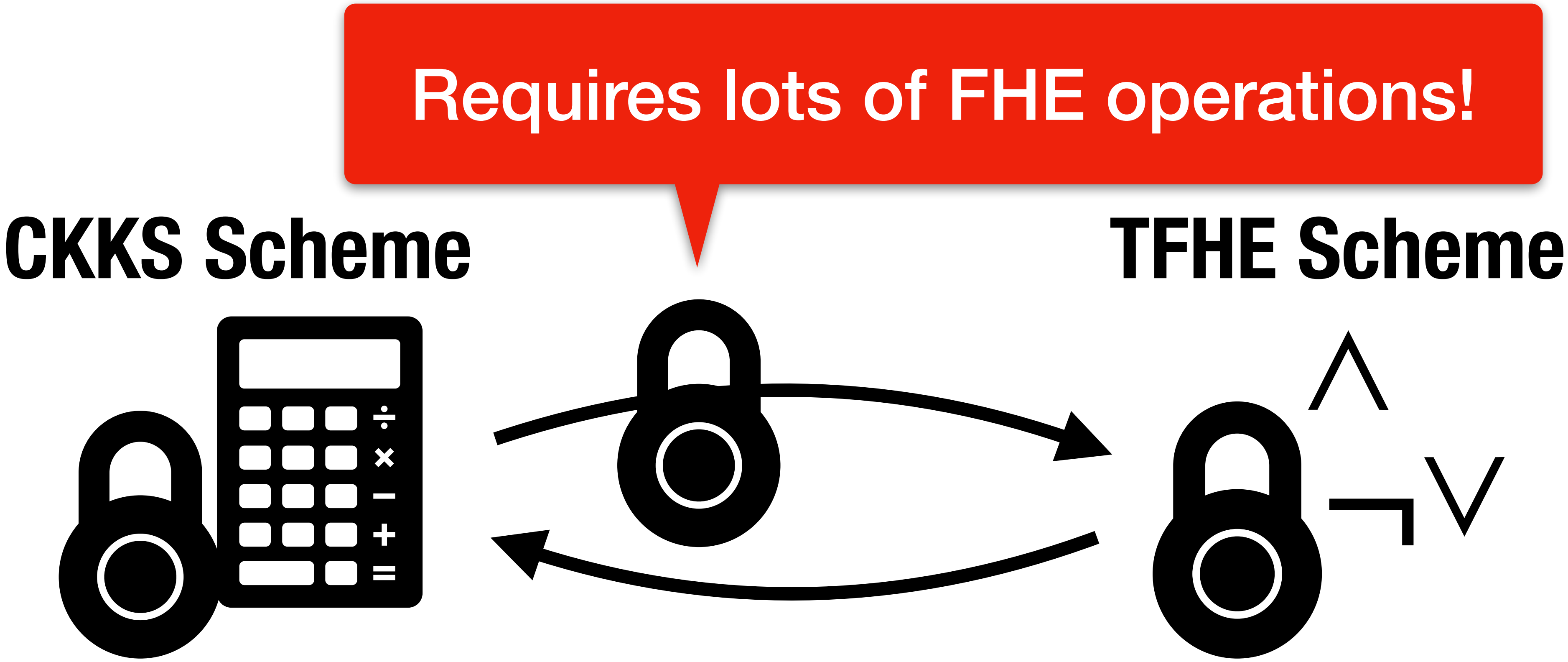
Challenge: Scheme switch. is slow

Scheme switching: (Essentially) homomorphic re-encryption



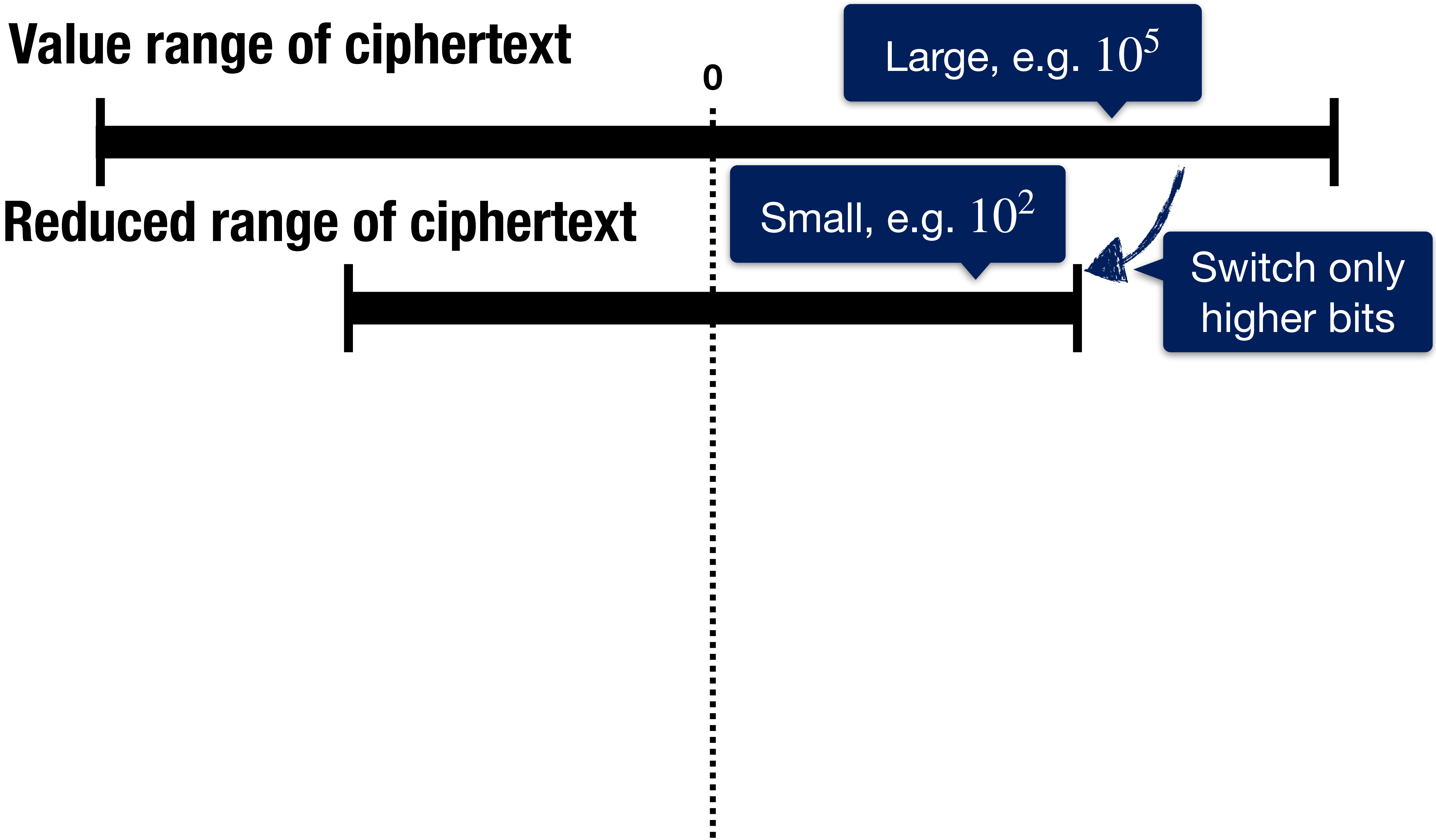
Challenge: Scheme switch. is slow

Scheme switching: (Essentially) homomorphic re-encryption

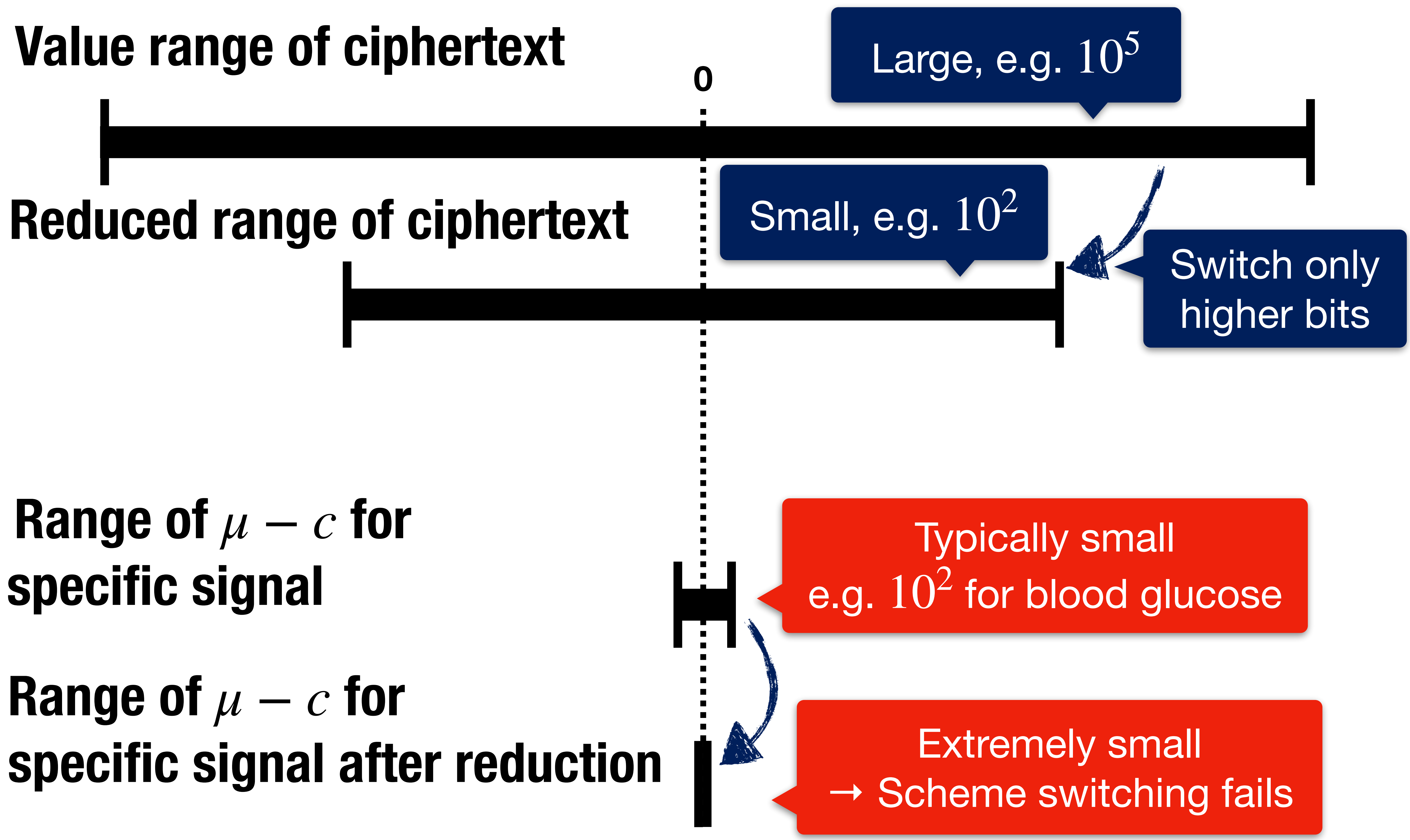


M. Waga (Kyoto U.)

Optimization: Partial scheme switch

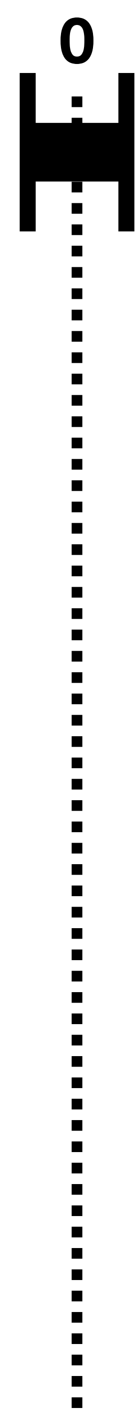


Optimization: Partial scheme switch



Optimization: Partial scheme switch

Range of $\mu - c$ for specific signal



Typically small
e.g. 10^2 for blood glucose

Optimization: Partial scheme switch

Range of $\mu - c$ for specific signal

0

Typically small
e.g. 10^2 for blood glucose

Amplify $\mu - c$
using signal range

Typically available from
domain knowledge

Range of $\mu - c$ for specific signal after scaling

Range of $\mu - c$ for specific signal after scaling & reduction

Large enough for
scheme switching

Complexity Analysis

Linear time and constant space wrt. log length
→ Scalable for online monitoring

	TFHE operations	CKKS Operations	Scheme Switching
Block	$O(n2^{\|\varphi\|})$	$O(n\|\varphi\|)$	$O(n\|\varphi\|)$
Reverse	$O(n2^{2^{\|\varphi\|}})$	$O(n\|\varphi\|)$	$O(n\|\varphi\|)$

New parts are linear to $|\varphi|$

Outline

- Review: Oblivious LTL monitoring with FHE
- Oblivious monitoring of STL with arith. predicates
  - Overview of the workflow
  - Optimization of scheme switching
- Experiments

Setting of Experiments

- Implemented in C++ based on (Banno, CAV'22)
  - Microsoft SEAL for CKKS, TFHEpp for TFHE
- Benchmarks: Blood Glucose (Banno, CAV'22) and RSS
- AWS EC2 c6i.2xlarge (8 Core 16 GiB RAM) with Ubuntu 22.04

Overall Results

	Block		Reverse	
	Exec. Time	DFA Size	Exec. Time	DFA Size
$BGLvl_1$	323 ms/value	703	659 ms/value	172,402
$BGLvl_2$	318 ms/value	703	660 ms/value	172,402
$BGLvl_4$	254 ms/value	703	**OOM**	**OOM**
$BGLvl_5$	458 ms/value	72,603	**OOM**	**OOM**
$BGLvl_6$	519 ms/value	72,603	**OOM**	**OOM**
$BGLvl_7$	393 ms/value	3	290 ms/value	3
$BGLvl_8$	384 ms/value	5	300 ms/value	5
$BGLvl_{10}$	363 ms/value	27	248 ms/value	27
$BGLvl_{11}$	346 ms/value	27	262 ms/value	27
RSS	569 ms/value	179	511 ms/value	218

Overall Results

Fast enough for Blood Glucose (typical sampling rate > 1 min.)

	Block		Reverse	
	Exec. Time	DFA Size	Exec. Time	DFA Size
$BGLvl_1$	323 ms/value	703	659 ms/value	172,402
$BGLvl_2$	318 ms/value	703	660 ms/value	172,402
$BGLvl_4$	254 ms/value	703	**OOM**	**OOM**
$BGLvl_5$	458 ms/value	72,603	**OOM**	**OOM**
$BGLvl_6$	519 ms/value	72,603	**OOM**	**OOM**
$BGLvl_7$	393 ms/value	3	290 ms/value	3
$BGLvl_8$	384 ms/value	5	300 ms/value	5
$BGLvl_{10}$	363 ms/value	27	248 ms/value	27
$BGLvl_{11}$	346 ms/value	27	262 ms/value	27
RSS	569 ms/value	179	511 ms/value	218

Overall Results

Fast enough for Blood Glucose (typical sampling rate > 1 min.)

	Block		Reverse	
	Exec. Time	DFA Size	Exec. Time	DFA Size
$BGLvl_1$	323 ms/value	703	659 ms/value	172,402
$BGLvl_2$	318 ms/value	703	660 ms/value	172,402
$BGLvl_4$	254 ms/value	703	**OOM**	**OOM**
$BGLvl_5$	458 ms/value	72,603	**OOM**	**OOM**
$BGLvl_6$	519 ms/value	72,603	**OOM**	**OOM**
$BGLvl_7$	393 ms/value	3	290 ms/value	3
$BGLvl_8$	384 ms/value	5	300 ms/value	5
$BGLvl_{10}$	363 ms/value	27	248 ms/value	27
$BGLvl_{11}$	346 ms/value	27	262 ms/value	27
RSS	569 ms/value	179	511 ms/value	218

Maybe acceptable for less safety critical usage, e.g. reducing traffic jam

Optimized Scheme Switching

Our optimization reduces exec. time about 30%

	Optimized		Naive	
	Block	Reverse	Block	Reverse
$BGLvl_1$	323 ms/value	659 ms/value	500 ms/value	850 ms/value
$BGLvl_2$	318 ms/value	660 ms/value	517 ms/value	858 ms/value
$BGLvl_4$	254 ms/value	OOM	455 ms/value	OOM
$BGLvl_5$	458 ms/value	OOM	636 ms/value	OOM
$BGLvl_6$	519 ms/value	OOM	650 ms/value	OOM
$BGLvl_7$	393 ms/value	290 ms/value	585 ms/value	501 ms/value
$BGLvl_8$	384 ms/value	300 ms/value	600 ms/value	505 ms/value
$BGLvl_{10}$	363 ms/value	248 ms/value	539 ms/value	423 ms/value
$BGLvl_{11}$	346 ms/value	262 ms/value	536 ms/value	419 ms/value
RSS	569 ms/value	511 ms/value	943 ms/value	789 ms/value

Comparison with (Banno et al., CAV'22)

Scheme switching is very slow
→ (Banno et al., CAV'22) is faster if it works

	Ours		(Banno et al., CAV'22)	
	Block	Reverse	Block	Reverse
$BGLvl_1$	323 ms/value	659 ms/value	102 ms/value	OOM
$BGLvl_2$	318 ms/value	660 ms/value	102 ms/value	OOM
$BGLvl_4$	254 ms/value	OOM	28.4 ms/value	OOM
$BGLvl_5$	458 ms/value	OOM	OOM	OOM
$BGLvl_6$	519 ms/value	OOM	OOM	OOM
$BGLvl_7$	393 ms/value	290 ms/value	95.0 ms/value	0.876 ms/value
$BGLvl_{10}$	363 ms/value	248 ms/value	111 ms/value	5.54 ms/value
$BGLvl_{11}$	346 ms/value	262 ms/value	114 ms/value	8.84 ms/value

Comparison with (Banno et al., CAV'22)

Scheme switching is very slow
→ (Banno et al., CAV'22) is faster if it works

	Ours		(Banno et al., CAV'22)	
	Block	Reverse	Block	Reverse
$BGLvl_1$	323 ms/value	659 ms/value	102 ms/value	OOM
$BGLvl_2$	318 ms/value	660 ms/value	102 ms/value	OOM
$BGLvl_4$	254 ms/value	OOM	28.4 ms/value	OOM
$BGLvl_5$	458 ms/value	OOM	OOM	OOM
$BGLvl_6$	519 ms/value	OOM	OOM	OOM
$BGLvl_7$	393 ms/value	290 ms/value	95.0 ms/value	0.876 ms/value
$BGLvl_{10}$	363 ms/value	248 ms/value	111 ms/value	5.54 ms/value
$BGLvl_{11}$	346 ms/value	262 ms/value	114 ms/value	8.84 ms/value

Comparison with (Banno et al., CAV'22)

Monitored value: numeric message instead of bit seq.
→ Smaller DFA

	Ours		(Banno et al., CAV'22)	
	Block DFA	Reverse DFA	Block DFA	Reverse DFA
$BGLvl_1$	703	172,402	10,524	**OOM**
$BGLvl_2$	703	172,402	11,126	**OOM**
$BGLvl_4$	703	**OOM**	7,026	**OOM**
$BGLvl_5$	72,603	**OOM**	**OOM**	**OOM**
$BGLvl_6$	72,603	**OOM**	**OOM**	**OOM**
$BGLvl_7$	3	3	21	20
$BGLvl_{10}$	27	27	237	237
$BGLvl_{11}$	27	27	390	390

Conclusions & Future Directions

- Online oblivious STL monitoring protocol
  - Combining CKKS and TFHE schemes
- Optimization of scheme switching for STL monitoring
- Implementation + experiments
  → Fast enough for blood glucose monitoring
  Works for RSS monitoring, too

Future directions

- Monitoring of *analog-digital mixed* signals
- Practical case study

Appendix

Ciphertexts for FHE: (R)LWE

Secret key | Raw plaintext

Ciphertext: (a, b) s.t. $b = s \cdot a + p + e$

Random "mask" | Random "noise"

- Random "noise" makes attack hard
- Each FHE operation amplifies noise
  e.g. × 2 by addition (on average)
- If $p + e \approx p$, we can decrypt correctly (at least approx.)
  → We need to keep noise small
  e.g., by Bootstrapping or using large ciphertext

Ciphertexts for FHE: (R)LWE

Secret key

Raw plaintext

Ciphertext: (a, b) s.t. $b = s \cdot a + p + e$

Random “mask”

Random “noise”

- Random “noise” makes attack hard
- Each FHE operation amplifies noise
  e.g. × 2 by addition (on average)
- If $p + e \approx p$, we can decrypt correctly (at least approx.)
  → We need to keep noise small
  e.g., by Bootstrapping or using large ciphertext

Formulas in Experiments

BGLvl_1	$\square_{[100,700]}(glucose \geq 70)$
BGLvl_2	$\square_{[100,700]}(glucose < 350)$
BGLvl_4	$\square_{[600,700]}(glucose < 200)$
BGLvl_5	$\neg\Diamond_{[200,600]}\square_{[0,180]}(glucose \geq 240)$
BGLvl_6	$\neg\Diamond_{[200,600]}\square_{[0,180]}(glucose < 70)$

BGLvl_7	$\square(glucose \geq 70 \wedge glucose < 180)$
BGLvl_8	$\square(\Delta glucose \geq -5 \wedge \Delta glucose < 3)$
BGLvl_{10}	$\square(glucose < 60 \Rightarrow \Diamond_{[0,25]} glucose \geq 60)$
BGLvl_{11}	$\square(glucose > 200 \Rightarrow \Diamond_{[0,25]} glucose < 200)$
RSS	$\square(S \wedge \mathcal{X}\neg S \Rightarrow \mathcal{X}(\varphi_{preBr} \wedge \varphi_{Br}))$

Client's Cost

Execution Time

	Enc. w/ public key [ms/value]	Enc. w/ private key [ms/value]	Decryption [ms/ciphertext]
NanoPi R6S (*w/* AES accelerator)	6.82	2.21	1.17×10^{-3}
Raspberry Pi 4 (*w/o* AES accelerator)	12.7	4.44	1.72×10^{-3}

Memory Usage

	Enc. w/ public key	Enc. w/ private key	Dec.
NanoPi R6S (*w/* AES accelerator)	360 656 kB	298 951.2 kB	6876.8 kB
Raspberry Pi 4 (*w/o* AES accelerator)	360 704 kB	299 089.6 kB	7168 kB

Detailed Experiment Results

	DFA eval. (sec.)				CKKS to TFHE (sec.)				CKKS eval. (sec.)				Runtime (sec.)			
	ArithHomFA$_{Opt}$		ArithHomFA$_{Naive}$		ArithHomFA$_{Opt}$		ArithHomFA$_{Naive}$		ArithHomFA$_{Opt}$		ArithHomFA$_{Naive}$		ArithHomFA$_{Opt}$		ArithHomFA$_{Naive}$	
	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse
BGLvl$_1$	6.09e+01	2.99e+02	6.10e+01	3.02e+02	1.72e+02	1.76e+02	2.99e+02	3.12e+02	5.77e-02	5.55e-02	5.51e-02	5.32e-02	2.33e+02	4.75e+02	3.60e+02	6.13e+02
BGLvl$_2$	6.02e+01	3.01e+02	6.48e+01	3.00e+02	1.69e+02	1.76e+02	3.08e+02	3.18e+02	7.29e-02	7.01e-02	7.30e-02	6.99e-02	2.29e+02	4.76e+02	3.73e+02	6.18e+02
BGLvl$_4$	1.23e+01	**OOM**	1.33e+01	**OOM**	1.70e+02	**OOM**	3.15e+02	**OOM**	8.59e-02	**OOM**	8.67e-02	**OOM**	1.83e+02	**OOM**	3.28e+02	**OOM**
BGLvl$_5$	1.57e+02	**OOM**	1.56e+02	**OOM**	1.73e+02	**OOM**	3.02e+02	**OOM**	5.67e-02	**OOM**	5.34e-02	**OOM**	3.30e+02	**OOM**	4.59e+02	**OOM**
BGLvl$_6$	1.87e+02	**OOM**	1.59e+02	**OOM**	1.87e+02	**OOM**	3.09e+02	**OOM**	8.22e-02	**OOM**	7.13e-02	**OOM**	3.74e+02	**OOM**	4.68e+02	**OOM**
BGLvl$_7$	9.79e+02	1.18e+01	9.56e+02	1.13e+01	2.98e+03	2.91e+03	4.94e+03	5.04e+03	1.65e+00	1.95e+00	1.62e+00	1.91e+00	3.96e+03	2.93e+03	5.90e+03	5.06e+03
BGLvl$_8$	9.63e+02	1.24e+01	9.67e+02	1.28e+01	2.91e+03	3.01e+03	5.08e+03	5.07e+03	2.49e+00	3.17e+00	2.60e+00	3.10e+00	3.87e+03	3.03e+03	6.05e+03	5.09e+03
BGLvl$_{10}$	1.15e+03	1.23e+01	1.14e+03	1.16e+01	2.51e+03	2.48e+03	4.29e+03	4.25e+03	8.26e-01	8.87e-01	8.11e-01	8.39e-01	3.66e+03	2.50e+03	5.43e+03	4.26e+03
BGLvl$_{11}$	1.09e+03	1.25e+01	1.14e+03	1.13e+01	2.40e+03	2.63e+03	4.26e+03	4.22e+03	7.79e-01	9.81e-01	7.84e-01	8.49e-01	3.49e+03	2.65e+03	5.40e+03	4.23e+03
RSS	4.89e+00	5.11e-01	5.10e+00	4.94e-01	2.25e+01	2.41e+01	4.07e+01	3.77e+01	4.61e-01	4.77e-01	4.79e-01	4.60e-01	2.79e+01	2.50e+01	4.62e+01	3.87e+01

Detailed Experiment Results

Depends on DFA size

Slow

Fast

	DFA eval. (sec.)				CKKS to TFHE (sec.)				CKKS eval. (sec.)				Runtime (sec.)			
	ArithHomFA$_{\text{Opt}}$		ArithHomFA$_{\text{Naive}}$		ArithHomFA$_{\text{Opt}}$		ArithHomFA$_{\text{Naive}}$		ArithHomFA$_{\text{Opt}}$		ArithHomFA$_{\text{Naive}}$		ArithHomFA$_{\text{Opt}}$		ArithHomFA$_{\text{Naive}}$	
	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse	Block	Reverse
BGLvl$_1$	6.09e+01	2.99e+02	6.10e+01	3.02e+02	1.72e+02	1.76e+02	2.99e+02	3.12e+02	5.77e-02	5.55e-02	5.51e-02	5.32e-02	2.33e+02	4.75e+02	3.60e+02	6.13e+02
BGLvl$_2$	6.02e+01	3.01e+02	6.48e+01	3.00e+02	1.69e+02	1.76e+02	3.08e+02	3.18e+02	7.29e-02	7.01e-02	7.30e-02	6.99e-02	2.29e+02	4.76e+02	3.73e+02	6.18e+02
BGLvl$_4$	1.23e+01	OOM	1.33e+01	OOM	1.70e+02	OOM	3.15e+02	OOM	8.59e-02	OOM	8.67e-02	OOM	1.83e+02	OOM	3.28e+02	OOM
BGLvl$_5$	1.57e+02	OOM	1.56e+02	OOM	1.73e+02	OOM	3.02e+02	OOM	5.67e-02	OOM	5.34e-02	OOM	3.30e+02	OOM	4.59e+02	OOM
BGLvl$_6$	1.87e+02	OOM	1.59e+02	OOM	1.87e+02	OOM	3.09e+02	OOM	8.22e-02	OOM	7.13e-02	OOM	3.74e+02	OOM	4.68e+02	OOM
BGLvl$_7$	9.79e+02	1.18e+01	9.56e+02	1.13e+01	2.98e+03	2.91e+03	4.94e+03	5.04e+03	1.65e+00	1.95e+00	1.62e+00	1.91e+00	3.96e+03	2.93e+03	5.90e+03	5.06e+03
BGLvl$_8$	9.63e+02	1.24e+01	9.67e+02	1.28e+01	2.91e+03	3.01e+03	5.08e+03	5.07e+03	2.49e+00	3.17e+00	2.60e+00	3.10e+00	3.87e+03	3.03e+03	6.05e+03	5.09e+03
BGLvl$_{10}$	1.15e+03	1.23e+01	1.14e+03	1.16e+01	2.51e+03	2.48e+03	4.29e+03	4.25e+03	8.26e-01	8.87e-01	8.11e-01	8.39e-01	3.66e+03	2.50e+03	5.43e+03	4.26e+03
BGLvl$_{11}$	1.09e+03	1.25e+01	1.14e+03	1.13e+01	2.40e+03	2.63e+03	4.26e+03	4.22e+03	7.79e-01	9.81e-01	7.84e-01	8.49e-01	3.49e+03	2.65e+03	5.40e+03	4.23e+03
RSS	4.89e+00	5.11e-01	5.10e+00	4.94e-01	2.25e+01	2.41e+01	4.07e+01	3.77e+01	4.61e-01	4.77e-01	4.79e-01	4.60e-01	2.79e+01	2.50e+01	4.62e+01	3.87e+01