ABE Squared: Accurately Benchmarking Efficiency of Attribute-Based Encryption

Antonio de la Piedra[1] **Marloes Venema**[2] Greg Alpár[2,3]

Kudelski Security Research Team, Cheseaux-sur-Lausanne, Switzerland

Radboud University, Nijmegen, the Netherlands

Open University of the Netherlands, Heerlen, the Netherlands

CHES 2022

Motivation

- Attribute-based encryption (ABE) is an advanced type of public-key encryption in which the keys are associated with attributes
- Various use cases, e.g., cloud-based settings
- Allows for enforcement of access control on a cryptographic level

Motivation

- Attribute-based encryption (ABE) is an advanced type of public-key encryption in which the keys are associated with attributes
- Various use cases, e.g., cloud-based settings
- Allows for enforcement of access control on a cryptographic level
- While many (pairing-based) schemes exist, few have working implementations
- Existing implementations may not be fairly comparable

Motivation

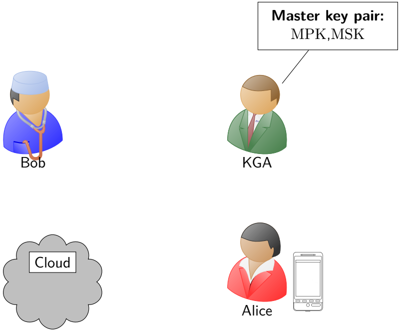
- Attribute-based encryption (ABE) is an advanced type of public-key encryption in which the keys are associated with attributes
- Various use cases, e.g., cloud-based settings
- Allows for enforcement of access control on a cryptographic level
- While many (pairing-based) schemes exist, few have working implementations
- Existing implementations may not be fairly comparable
- **Our goal:** accurately benchmarking and comparing schemes, efficiency analysis, new speed records

High-level overview

High-level overview

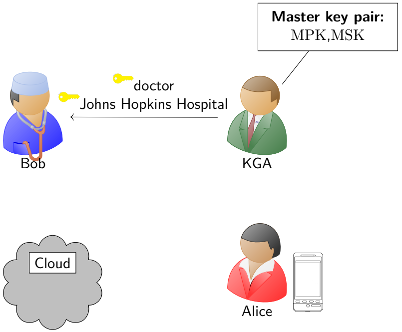
Ciphertext-policy attribute-based encryption (CP-ABE)

Setup:



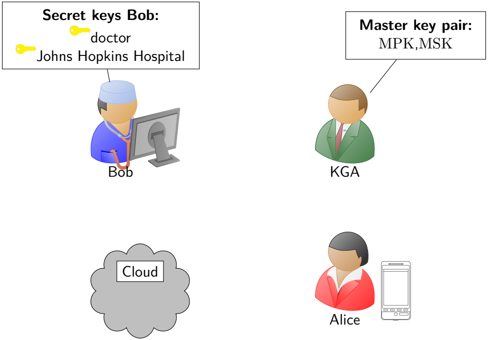
Ciphertext-policy attribute-based encryption (CP-ABE)

Key generation:



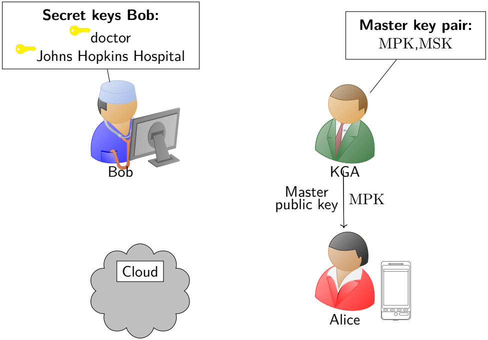
Ciphertext-policy attribute-based encryption (CP-ABE)

Key generation:



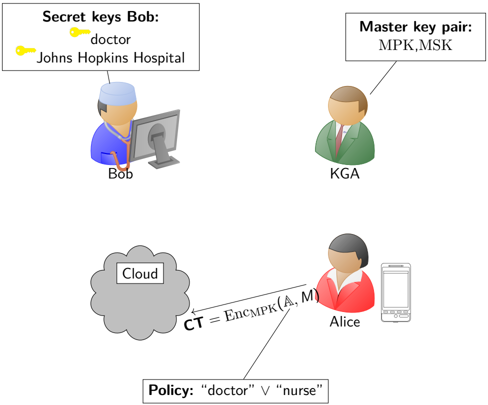
Ciphertext-policy attribute-based encryption (CP-ABE)

Encryption:

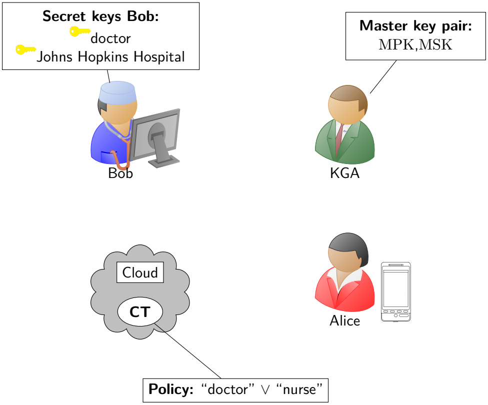


Ciphertext-policy attribute-based encryption (CP-ABE)

Encryption:

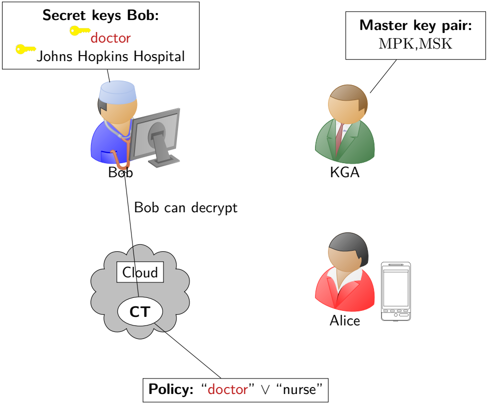


Ciphertext-policy attribute-based encryption (CP-ABE)



Ciphertext-policy attribute-based encryption (CP-ABE)

Decryption:



Pairing-based ABE

- We focus on pairing-based ABE
- Most established: many desirable practical properties, high security guarantees and efficient

Pairing-based ABE

- We focus on pairing-based ABE
- Most established: many desirable practical properties, high security guarantees and efficient
- Unfortunately, not post-quantum secure

Pairing-based ABE

- We focus on pairing-based ABE
- Most established: many desirable practical properties, high security guarantees and efficient
- Unfortunately, not post-quantum secure
- Post-quantum secure schemes exist
- However, still heavily under development, e.g., to achieve the same security guarantees and other desirable properties

High-level overview

Benchmarking crypto

Usually:

- Make some choices (e.g., architecture, CPU, platform) required for a fair comparison
- Implement and optimize with a strategy in mind

Benchmarking crypto

Usually:

- Make some choices (e.g., architecture, CPU, platform) required for a fair comparison
- Implement and optimize with a strategy in mind

Typically, in ABE:

- Choose a framework for rapid prototyping, e.g., Charm [AGM+13]
- Implement, maybe optimize some parts

General problems in implementing pairing-based ABE

Many variables that need to be optimized, e.g.,

General problems in implementing pairing-based ABE

Many variables that need to be optimized, e.g.,

- access policies
- arithmetic and group operations
- many different pairing-friendly groups
- type conversion

General problems in implementing pairing-based ABE

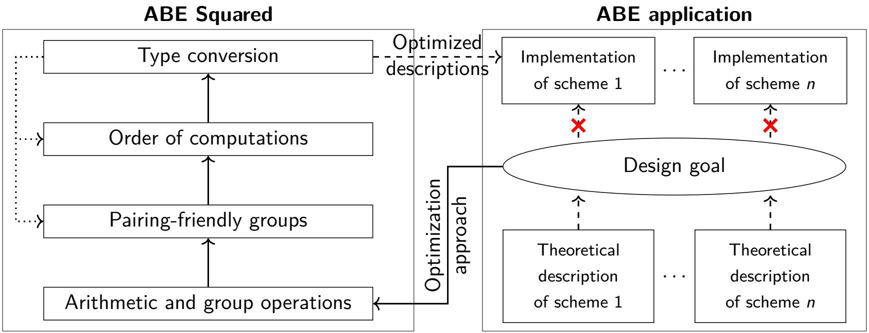
Many variables that need to be optimized, e.g.,

- access policies
- arithmetic and group operations
- many different pairing-friendly groups
- type conversion

Some of these really depend on what the designer tries to optimize, e.g., the key generation for a KGA who receives many key requests

High-level overview

Overview of ABE Squared



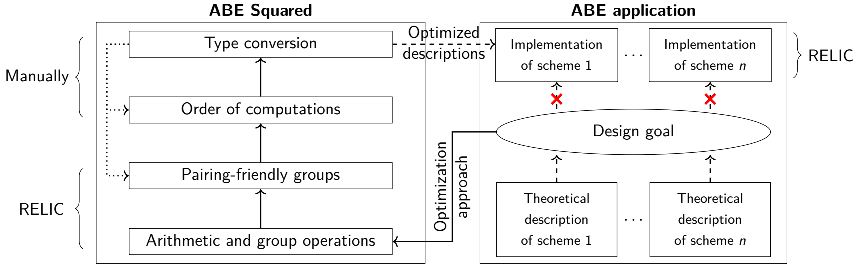
The arrows have the following meaning:

$a \longrightarrow b$ = "a influences b"

$a \dashrightarrow b$ = "a may require adjustment in b"

$a \dashrightarrow b$ = "a is input to b"/"b is output of a"

Overview of ABE Squared (continued)



The arrows have the following meaning:

$a \longrightarrow b$ = "a influences b"

$a \cdots\rightarrow b$ = "a may require adjustment in b"

$a \dashrightarrow b$ = "a is input to b"/"b is output of a"

New heuristics

- Interaction between the upper two layers and how they interact with the lower two layers has not been investigated sufficiently for ABE yet

New heuristics

- Interaction between the upper two layers and how they interact with the lower two layers has not been investigated sufficiently for ABE yet
- Previous type-conversion methods typically only allow for optimized key or ciphertext sizes
- These cannot be used to optimize decryption

New heuristics

- Interaction between the upper two layers and how they interact with the lower two layers has not been investigated sufficiently for ABE yet
- Previous type-conversion methods typically only allow for optimized key or ciphertext sizes
- These cannot be used to optimize decryption
- We provide manual heuristics that take the interactions between the different layers into account
- Allows us to better optimize e.g., the decryption algorithm than previous methods allow

Optimized type conversion

Pairing: $e\colon \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$, where $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ are groups of order p.

Optimized type conversion

Pairing: $e\colon \mathbb{G} \times \mathbb{H} \rightarrow \mathbb{G}_T$, where $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ are groups of order p.

Theory: often assumes $\mathbb{G} = \mathbb{H}$
Practice: $\mathbb{G} \neq \mathbb{H}$

Optimized type conversion

Pairing: $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ are groups of order p.

Theory: often assumes $\mathbb{G} = \mathbb{H}$
Practice: $\mathbb{G} \neq \mathbb{H}$ $\Rightarrow$ efficiency in $\mathbb{G} \neq$ efficiency in $\mathbb{H}$

Optimized type conversion

Pairing: $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ are groups of order p.

Theory: often assumes $\mathbb{G} = \mathbb{H}$
Practice: $\mathbb{G} \neq \mathbb{H}$ $\Rightarrow$ efficiency in $\mathbb{G} \neq$ efficiency in $\mathbb{H}$

Type conversion: convert the scheme from setting with $\mathbb{G} = \mathbb{H}$ to $\mathbb{G} \neq \mathbb{H}$.

Optimized type conversion

Pairing: $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ are groups of order p.

Theory: often assumes $\mathbb{G} = \mathbb{H}$
Practice: $\mathbb{G} \neq \mathbb{H}$ $\Rightarrow$ efficiency in $\mathbb{G}$ $\neq$ efficiency in $\mathbb{H}$

Type conversion: convert the scheme from setting with $\mathbb{G} = \mathbb{H}$ to $\mathbb{G} \neq \mathbb{H}$.

Many ways to instantiate the scheme in $\mathbb{G}$ and $\mathbb{H}$!

Optimized type conversion

Pairing: $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ are groups of order p.

Theory: often assumes $\mathbb{G} = \mathbb{H}$
Practice: $\mathbb{G} \neq \mathbb{H}$ $\Rightarrow$ efficiency in $\mathbb{G} \neq$ efficiency in $\mathbb{H}$

Type conversion: convert the scheme from setting with $\mathbb{G} = \mathbb{H}$ to $\mathbb{G} \neq \mathbb{H}$.

Many ways to instantiate the scheme in $\mathbb{G}$ and $\mathbb{H}$!

Each instantiation performs differently.

Optimized type conversion

Pairing: $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ are groups of order p.

Theory: often assumes $\mathbb{G} = \mathbb{H}$
Practice: $\mathbb{G} \neq \mathbb{H}$ $\Rightarrow$ efficiency in $\mathbb{G} \neq$ efficiency in $\mathbb{H}$

Type conversion: convert the scheme from setting with $\mathbb{G} = \mathbb{H}$ to $\mathbb{G} \neq \mathbb{H}$.

Many ways to instantiate the scheme in $\mathbb{G}$ and $\mathbb{H}$!

Each instantiation performs differently.

Most efficient one depends on the lower three layers, i.e., arithmetic and group operations, chosen group and the order of the group operations.

Optimized type conversion

Pairing: $e\colon \mathbb{G} \times \mathbb{H} \to \mathbb{G}_T$, where $\mathbb{G}, \mathbb{H}$ and $\mathbb{G}_T$ are groups of order p.

Theory: often assumes $\mathbb{G} = \mathbb{H}$
Practice: $\mathbb{G} \neq \mathbb{H}$ $\Rightarrow$ efficiency in $\mathbb{G} \neq$ efficiency in $\mathbb{H}$

Type conversion: convert the scheme from setting with $\mathbb{G} = \mathbb{H}$ to $\mathbb{G} \neq \mathbb{H}$.

Many ways to instantiate the scheme in $\mathbb{G}$ and $\mathbb{H}$!

Each instantiation performs differently.

Most efficient one depends on the lower three layers, i.e., arithmetic and group operations, chosen group and the order of the group operations.

Our heuristics: find most efficient instantiation in $\mathbb{G}$ and $\mathbb{H}$ given a specific design goal. (Also depends on the chosen group!)

High-level overview

Benchmarks for differently optimized schemes

Implementation of Wat11-I in RELIC, on the BLS12-381 curve, based on their optimization approaches[1] (OA). The costs are expressed in 10^3 clock cycles[2].

OA	Key generation				Encryption				Decryption			
	# of attributes				# of attributes				# of attributes			
	1	10	100	Increase	1	10	100	Increase	1	10	100	Increase
OE & OD	759	3029	25653	143.0%	**990**	**4540**	**39951**	-	**2005**	**7379**	**58515**	-
OK	**317**	**1249**	**10555**	-	1756	10814	101181	153.3%	2016	7611	63151	7.9%

[1] OE/OD/OK = optimized encryption/decryption/key generation.

[2] AMD Ryzen 7 PRO 4750 processor, one single core at 4.1 GHz.

Benchmarks for differently optimized schemes

Implementation of Wat11-I in RELIC, on the BLS12-381 curve, based on their optimization approaches[1] (OA). The costs are expressed in 10^3 clock cycles[2].

OA	Key generation				Encryption				Decryption			
	# of attributes				# of attributes				# of attributes			
	1	10	100	Increase	1	10	100	Increase	1	10	100	Increase
OE & OD	759	3029	25653	143.0%	**990**	**4540**	**39951**	-	**2005**	**7379**	**58515**	-
OK	**317**	**1249**	**10555**	-	1756	10814	101181	153.3%	2016	7611	63151	7.9%

- Design goal influences the type conversion
- e.g., it yields a difference of a factor of ≈ 2.5 in computational costs for BLS12-381

[1] OE/OD/OK = optimized encryption/decryption/key generation.

[2] AMD Ryzen 7 PRO 4750 processor, one single core at 4.1 GHz.

Performance analysis

To demonstrate the framework, we have implemented and benchmarked three schemes with the same practical properties (achieved in different ways) in RELIC:

- Wat11-IV [Wat11]: implemented in libraries such as Charm and OpenABE
- RW13 [RW13]: implemented in Charm, outperformed by Wat11-IV
- AC17-LU [AC17]: not implemented

Many follow-up works build on these schemes and are structurally similar.

Benchmarks

OA	Scheme	Curve	Key generation		Encryption		Decryption	
			Costs	Increase %	Costs	Increase %	Costs	Increase %
OE	Wat11-IV	BLS12-381	42275	0.2%	77641	48.8%	58290	543.4%
	RW13	BLS12-381	51401	21.8%	54491	4.4%	112072	1137.1%
	AC17-LU	BLS12-381	**42196**	-	**52176**	-	**9060**	-
OK	Wat11-IV	BLS12-381	42135	94.6%	77898	48.9%	58441	543.9%
	RW13	BLS12-381	**21657**	-	128221	145.0%	118998	1211.2%
	AC17-LU	BLS12-381	41913	93.5%	**52326**	-	**9076**	-
OD	Wat11-IV	BLS12-381	**42275**	-	77641	42.5%	58290	1336.5%
	RW13	BLS12-381	51401	21.6%	**54491**	-	112072	2661.9%
	AC17-LU	BN382	45093	6.7%	59276	8.8%	**4058**	-

Benchmarks

OA	Scheme	Curve	Key generation		Encryption		Decryption	
			Costs	Increase %	Costs	Increase %	Costs	Increase %
OE	Wat11-IV	BLS12-381	42275	0.2%	77641	48.8%	58290	543.4%
	RW13	BLS12-381	51401	21.8%	54491	4.4%	112072	1137.1%
	AC17-LU	BLS12-381	**42196**	-	**52176**	-	**9060**	-
OK	Wat11-IV	BLS12-381	42135	94.6%	77898	48.9%	58441	543.9%
	RW13	BLS12-381	**21657**	-	128221	145.0%	118998	1211.2%
	AC17-LU	BLS12-381	41913	93.5%	**52326**	-	**9076**	-
OD	Wat11-IV	BLS12-381	**42275**	-	77641	42.5%	58290	1336.5%
	RW13	BLS12-381	51401	21.6%	**54491**	-	112072	2661.9%
	AC17-LU	BN382	45093	6.7%	59276	8.8%	**4058**	-

- For an optimized encryption or decryption, use AC17-LU
- For an optimized key generation, use RW13

Benchmarks

OA	Scheme	Curve	Key generation		Encryption		Decryption	
			Costs	Increase %	Costs	Increase %	Costs	Increase %
OE	Wat11-IV	BLS12-381	42275	0.2%	77641	48.8%	58290	543.4%
	RW13	BLS12-381	51401	21.8%	54491	4.4%	112072	1137.1%
	AC17-LU	BLS12-381	**42196**	-	**52176**	-	**9060**	-
OK	Wat11-IV	BLS12-381	42135	94.6%	77898	48.9%	58441	543.9%
	RW13	BLS12-381	**21657**	-	128221	145.0%	118998	1211.2%
	AC17-LU	BLS12-381	41913	93.5%	**52326**	-	**9076**	-
OD	Wat11-IV	BLS12-381	**42275**	-	77641	42.5%	58290	1336.5%
	RW13	BLS12-381	51401	21.6%	**54491**	-	112072	2661.9%
	AC17-LU	BN382	45093	6.7%	59276	8.8%	**4058**	-

- For an optimized encryption or decryption, use AC17-LU
- For an optimized key generation, use RW13
- **Surprising result:** RW13 outperforms Wat11-IV in the key generation and encryption algorithms

High-level overview

Conclusion

- ABE Squared: a framework for accurately benchmarking efficiency of attribute-based encryption

Conclusion

- ABE Squared: a framework for accurately benchmarking efficiency of attribute-based encryption
- Aims to optimize ABE schemes for some chosen design goal by considering four optimization layers:
  - arithmetic and group operations
  - pairing-friendly group
  - order of the computations
  - type conversion

Conclusion

- ABE Squared: a framework for accurately benchmarking efficiency of attribute-based encryption
- Aims to optimize ABE schemes for some chosen design goal by considering four optimization layers:
  - arithmetic and group operations
  - pairing-friendly group
  - order of the computations
  - type conversion
- By optimizing multiple schemes with respect to the same goal, they can be compared more fairly

Conclusion

- ABE Squared: a framework for accurately benchmarking efficiency of attribute-based encryption
- Aims to optimize ABE schemes for some chosen design goal by considering four optimization layers:
  - arithmetic and group operations
  - pairing-friendly group
  - order of the computations
  - type conversion
- By optimizing multiple schemes with respect to the same goal, they can be compared more fairly
- Existing open-source libraries providing ABE implementations, e.g., Charm, OpenABE, can greatly benefit from our heuristics
- **Design goals matter:** for different goals, different schemes may perform the best

Questions?

Thank you for your attention!

- Our paper:
  - TCHES: tches.iacr.org/index.php/TCHES/article/view/9486
  - eprint: https://eprint.iacr.org/2022/038
- Our code: https://github.com/abecryptools/abe_squared

References I

[AC17] S. Agrawal and M. Chase.
Simplifying design and analysis of complex predicate encryption schemes.
In J.-S. Coron and J. B. Nielsen, editors, *EUROCRYPT*, volume 10210 of *Lecture Notes in Computer Science*, pages 627–656. Springer, 2017.

[AGM+13] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin.
Charm: a framework for rapidly prototyping cryptosystems.
J. Cryptogr. Eng., 3(2):111–128, 2013.

[RW13] Y. Rouselakis and B. Waters.
Practical constructions and new proof methods for large universe attribute-based encryption.
In A.-R. Sadeghi, V. D. Gligor, and M. Yung, editors, *CCS*, pages 463–474. ACM, 2013.

[Wat11] B. Waters.
Ciphertext-policy attribute-based encryption - an expressive, efficient, and provably secure realization.
In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, editors, *PKC*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2011.