Implementing pairing-based protocols

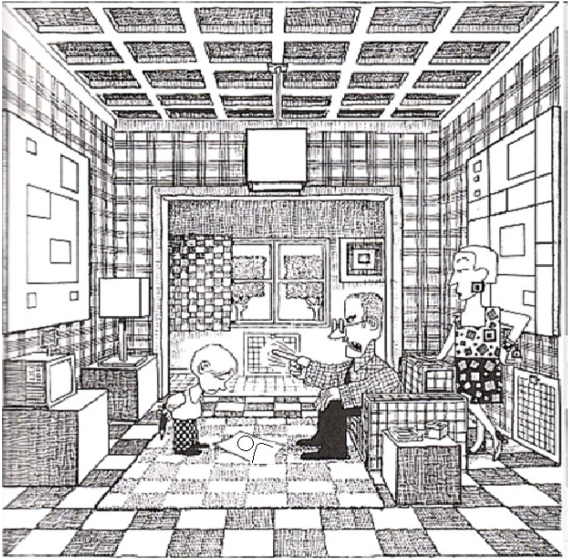
Francisco Rodríguez-Henríquez
CINVESTAV-IPN

Joint work with:

Luis J Dominguez Perez	CINVESTAV-LTI, México
Shigeo Mitsunari	Cybozu Labs, Japan
Ana H. Sánchez-Ramírez	CINVESTAV-IPN, México
Tadanori Teruya	RISEC-AIST, Japan
Eric Zavattoni	Université Claude Bernard, Lyon 1, France

The 6th International Conference on Pairing-Based Cryptography (Pairing 2013)
November 23, 2013

Elliptic curves



borrowed from Quino.

Elliptic curves

- An elliptic curve E defined over a prime field K, with characteristic different than 2,3, is specified by the simplified Weierstraß equation:

$$E/K : y^2 = x^3 + Ax + B$$

Elliptic curves

- An elliptic curve E defined over a prime field K, with characteristic different than 2,3, is specified by the simplified Weierstraß equation:

$$E/K : y^2 = x^3 + Ax + B$$

- Let us denote by $E(K)$ the set of rational points over the field K. Together with the point at infinity and an additive group law, $E(K)$ forms an abelian group

Elliptic curves

- An elliptic curve E defined over a prime field K, with characteristic different than 2,3, is specified by the simplified Weierstraß equation:

$$E/K : y^2 = x^3 + Ax + B$$

- Let us denote by $E(K)$ the set of rational points over the field K. Together with the point at infinity and an additive group law, $E(K)$ forms an abelian group
- The number of points in the curve is denoted as $\#E$, and the integer $t = p + 1 - \#E$, known as the trace of Frobenius, satisfies $|t| \leq 2\sqrt{p}$

Elliptic curves

- An elliptic curve E defined over a prime field K, with characteristic different than 2,3, is specified by the simplified Weierstraß equation:

$$E/K : y^2 = x^3 + Ax + B$$

- Let us denote by $E(K)$ the set of rational points over the field K. Together with the point at infinity and an additive group law, $E(K)$ forms an abelian group
- The number of points in the curve is denoted as $\#E$, and the integer $t = p + 1 - \#E$, known as the trace of Frobenius, satisfies $|t| \leq 2\sqrt{p}$
- Let r be a large prime with $r \mid \#E(\mathbb{F}_p)$ and $\gcd(r, p) = 1$. The embedding degree k is the smallest positive integer such that $r \mid (p^k - 1)$

Discrete logarithm cryptography

- $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = r$

Discrete logarithm cryptography

- $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = r$
- P, Q, are generators of the groups: $\mathbb{G}_1 = \langle P \rangle$, $\mathbb{G}_2 = \langle Q \rangle$

Discrete logarithm cryptography

- $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = r$
- P, Q, are generators of the groups: $\mathbb{G}_1 = \langle P \rangle$, $\mathbb{G}_2 = \langle Q \rangle$
- Scalar multiplication: for any integer n, we have $nP = \underbrace{P + P + \cdots + P}_{n-1 \text{ additions}}$

Discrete logarithm cryptography

- $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = r$
- P, Q, are generators of the groups: $\mathbb{G}_1 = \langle P \rangle$, $\mathbb{G}_2 = \langle Q \rangle$
- Scalar multiplication: for any integer n, we have $nP = \underbrace{P + P + \cdots + P}_{n-1 \text{ additions}}$
- Elliptic curve discrete logarithm problem: Given an elliptic curve E defined over a field $\mathbb{F}_q$ and $P, R \in E(\mathbb{F}_{q^k})$, find an integer n (if one exists) such that, $nP = R$

Discrete logarithm cryptography

- $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, additively-written cyclic groups of prime order $\#\mathbb{G}_1 = \#\mathbb{G}_2 = r$
- P, Q, are generators of the groups: $\mathbb{G}_1 = \langle P \rangle$, $\mathbb{G}_2 = \langle Q \rangle$
- Scalar multiplication: for any integer n, we have $nP = \underbrace{P + P + \cdots + P}_{n-1 \text{ additions}}$
- Elliptic curve discrete logarithm problem: Given an elliptic curve E defined over a field $\mathbb{F}_q$ and $P, R \in E(\mathbb{F}_{q^k})$, find an integer n (if one exists) such that, $nP = R$
- We assume that the discrete logarithm problem (DLP) in $\mathbb{G}_1$ is hard

Bilinear pairings

borrowed from Quino.

Bilinear pairing, basic definitions and properties (1/2)

Bilinear pairings were introduced by the French mathematician André Weil in 1940 under the name of couplages. Here, we define a bilinear pairing, or pairing for short, as a non-degenerate bilinear mapping,

$$\hat{e} : \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \ \mathbb{G}_T,$$

where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, are finite cyclic groups of prime order r. Pairings are classified according to the structure of their underlying groups.

Bilinear pairing, basic definitions and properties (1/2)

Bilinear pairings were introduced by the French mathematician André Weil in 1940 under the name of couplages. Here, we define a bilinear pairing, or pairing for short, as a non-degenerate bilinear mapping,

$$\hat{e} : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_T,$$

where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, are finite cyclic groups of prime order r. Pairings are classified according to the structure of their underlying groups.

- When $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is said to be of Type 1 (also called symmetric pairing);

Bilinear pairing, basic definitions and properties (1/2)

Bilinear pairings were introduced by the French mathematician André Weil in 1940 under the name of couplages. Here, we define a bilinear pairing, or pairing for short, as a non-degenerate bilinear mapping,

$$\hat{e} : \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T,$$

where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, are finite cyclic groups of prime order r. Pairings are classified according to the structure of their underlying groups.

- When $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is said to be of Type 1 (also called symmetric pairing);
- Otherwise, if $\mathbb{G}_1 \neq \mathbb{G}_2$, and no efficient computable homomorphism to map elements between these two groups is known, the pairing is said to be of Type 3 (also called asymmetric pairing)

Bilinear pairing, basic definitions and properties (1/2)

Bilinear pairings were introduced by the French mathematician André Weil in 1940 under the name of couplages. Here, we define a bilinear pairing, or pairing for short, as a non-degenerate bilinear mapping,

$$\hat{e} : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_T,$$

where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, are finite cyclic groups of prime order r. Pairings are classified according to the structure of their underlying groups.

- When $\mathbb{G}_1 = \mathbb{G}_2$, the pairing is said to be of Type 1 (also called symmetric pairing);
- Otherwise, if $\mathbb{G}_1 \neq \mathbb{G}_2$, and no efficient computable homomorphism to map elements between these two groups is known, the pairing is said to be of Type 3 (also called asymmetric pairing)

Note: There exist also Type 2 and Type 4 pairings. Type 2 pairings can be safely ignored [Chatterjee-Menezes D. Appl. Math 2011]

Bilinear pairing, basic definitions and properties (2/2)

A pairing is non-degenerate iff $\hat{e}(Q, P) \neq 1_{\mathbb{G}_T}$. The most important property of a pairing is its bilinearity, denoted as:

$$\begin{aligned} \hat{e}(Q_1 + Q_2, P) &= \hat{e}(Q_1, P) \cdot \hat{e}(Q_2, P); \\ \hat{e}(Q, P_1 + P_2) &= \hat{e}(Q, P_1) \cdot \hat{e}(Q, P_2). \end{aligned}$$

where $P_1, P_2 \in \mathbb{G}_1$; $Q_1, Q_2 \in \mathbb{G}_2$, and the result is in $\mathbb{G}_T$. From the above property, it follows that for any two integers n_1 and n_2,

$$\hat{e}(n_1 Q, n_2 P) = \hat{e}(n_1 n_2 Q, P) = \hat{e}(Q, n_1 n_2 P) = \hat{e}(Q, P)^{n_1 n_2}$$

Bilinear pairing, basic definitions and properties (2/2)

A pairing is non-degenerate iff $\hat{e}(Q, P) \neq 1_{\mathbb{G}_T}$. The most important property of a pairing is its bilinearity, denoted as:

$$\begin{aligned} \hat{e}(Q_1 + Q_2, P) &= \hat{e}(Q_1, P) \cdot \hat{e}(Q_2, P); \\ \hat{e}(Q, P_1 + P_2) &= \hat{e}(Q, P_1) \cdot \hat{e}(Q, P_2). \end{aligned}$$

where $P_1, P_2 \in \mathbb{G}_1$; $Q_1, Q_2 \in \mathbb{G}_2$, and the result is in $\mathbb{G}_T$. From the above property, it follows that for any two integers n_1 and n_2,

$$\hat{e}(n_1 Q, n_2 P) = \hat{e}(n_1 n_2 Q, P) = \hat{e}(Q, n_1 n_2 P) = \hat{e}(Q, P)^{n_1 n_2}$$

- Intuitively, scalar multiplication in $\mathbb{G}_2$ is much more expensive than in $\mathbb{G}_1$, hence, it is wise to place such operation in the latter group

Bilinear pairing, basic definitions and properties (2/2)

A pairing is non-degenerate iff $\hat{e}(Q, P) \neq 1_{\mathbb{G}_T}$. The most important property of a pairing is its bilinearity, denoted as:

$$\begin{aligned} \hat{e}(Q_1 + Q_2, P) &= \hat{e}(Q_1, P) \cdot \hat{e}(Q_2, P); \\ \hat{e}(Q, P_1 + P_2) &= \hat{e}(Q, P_1) \cdot \hat{e}(Q, P_2). \end{aligned}$$

where $P_1, P_2 \in \mathbb{G}_1$; $Q_1, Q_2 \in \mathbb{G}_2$, and the result is in $\mathbb{G}_T$. From the above property, it follows that for any two integers n_1 and n_2,

$$\hat{e}(n_1 Q, n_2 P) = \hat{e}(n_1 n_2 Q, P) = \hat{e}(Q, n_1 n_2 P) = \hat{e}(Q, P)^{n_1 n_2}$$

- Intuitively, scalar multiplication in $\mathbb{G}_2$ is much more expensive than in $\mathbb{G}_1$, hence, it is wise to place such operation in the latter group
- It is also believed that an exponentiation in $\mathbb{G}_T$ is cheaper than a pairing computation. Thus, some protocol designers may try to exploit this too

The rise of pairing-based cryptography: 20 years of history

- September 1993: The Menezes-Okamoto-Vanstone (MOV) attack polynomially reduces elliptic curve discrete problems into a discrete problem over a finite extension field using the Weil pairing
- April 1994: Frey and Rück introduced the Tate pairing in cryptography to carry out an attack similar to the MOV one
- January 2000: Sakai-Ohgishi-Kasahara discovered constructive properties of pairings (identity-based key exchange)
- July 2000: Joux presented a one round protocol for tripartite Diffie-Hellman
- August 2001: Boneh and Franklin proposed identity-based encryption using the Weil pairing [4800+ citations]
- December 2001: Boneh, Lynn and Shacham presented short signatures using the Weil pairing[1900+ citations]

Why pairing-based cryptography was universally accepted from the very beginning?

- Some of the constructions allow to solve emblematic problems in an elegant way

Why pairing-based cryptography was universally accepted from the very beginning?

- Some of the constructions allow to solve emblematic problems in an elegant way
- Pairings are a fascinating tool for both, number theory and protocol design disciplines

Why pairing-based cryptography was universally accepted from the very beginning?

- Some of the constructions allow to solve emblematic problems in an elegant way
- Pairings are a fascinating tool for both, number theory and protocol design disciplines
- Since the first proposed schemes, pairing-based protocols were presented in the language of provable security

Why pairing-based cryptography was universally accepted from the very beginning?

- Some of the constructions allow to solve emblematic problems in an elegant way
- Pairings are a fascinating tool for both, number theory and protocol design disciplines
- Since the first proposed schemes, pairing-based protocols were presented in the language of provable security
- Excellent timing: Intense debates about the usage of RSA Vs ECC were mostly over after 2000

Why pairing-based cryptography was universally accepted from the very beginning?

- Some of the constructions allow to solve emblematic problems in an elegant way
- Pairings are a fascinating tool for both, number theory and protocol design disciplines
- Since the first proposed schemes, pairing-based protocols were presented in the language of provable security
- Excellent timing: Intense debates about the usage of RSA Vs ECC were mostly over after 2000
- Many crypto conferences emerged during the last decade, including Pairing (back in 2007)

Pairing-based cryptography from the protocol design perspective

borrowed from Quino.

Pairing-based cryptography from the protocol design perspective

- As of today, hundreds of protocols have been proposed

Pairing-based cryptography from the protocol design perspective

- As of today, hundreds of protocols have been proposed
- a bilinear pairing is often seen as a black box that provides the bilinear property

Pairing-based cryptography from the protocol design perspective

- As of today, hundreds of protocols have been proposed
- a bilinear pairing is often seen as a black box that provides the bilinear property
- Most protocols use symmetric pairings

Pairing-based cryptography from the protocol design perspective

- As of today, hundreds of protocols have been proposed
- a bilinear pairing is often seen as a black box that provides the bilinear property
- Most protocols use symmetric pairings
- There is no good notion of the individual costs of the main cryptographic blocks within a protocol. As a consequence some protocol designers have a bad intuition of the costs associated to their schemes

A recent example of a practical pairing-based protocol

In a 2013 JoC paper, Lu-Ostrovsky-Sahai-Shacham-Waters presented three provably-secure aggregate signature, multisignature, and encrypted signature schemes.

A recent example of a practical pairing-based protocol

In a 2013 JoC paper, Lu-Ostrovsky-Sahai-Shacham-Waters presented three provably-secure aggregate signature, multisignature, and encrypted signature schemes.
In their paper the authors give the following concluding remark,

> *"In this paper we gave the first aggregate signature scheme which is provably secure without random oracles; the first multisignature scheme which is provably secure without random oracles; and the first verifiably encrypted signature scheme which is provably secure without random oracles [...] All our constructions are quite practical."*

A recent example of a practical pairing-based protocol

Sequential Aggregate Signatures, Multisignatures, and Verifiably Encrypted Signatures 369

Table 1. Comparison of aggregate signature schemes. Signatures are by l signers; k is the output length of a collision resistant hash function; "R.O." denotes if the security proof uses random oracles. Neven's scheme supports message recovery, which can reduce the effective signature overhead.

Scheme	R.O.	Sequential	Key model	Sig. size	Key size	Verification	Signing
BGLS	YES	NO	Chosen	160 bits	1920 bits	$l+1$ pair.	1 exp.
LMRS-1	YES	YES	Chosen	1024 bits	2048 bits	$2l$ exp.	verify + 1 exp.
LMRS-2	YES	YES	Registered	1024 bits	1024 bits	$4l$ mult.	verify + 1 exp.
Neven	YES	YES	Chosen	1184 bits	1024 bits	$2l$ mult.	verify + 1 exp.
Ours	NO	YES	Registered	320 bits	311040 bits	2 pair., $lk/2$ mult.	verify + 1 exp.

Credits: Material taken from the Journal of Cryptology, Springer

A recent example of a practical pairing-based protocol

Sequential Aggregate Signatures, Multisignatures, and Verifiably Encrypted Signatures 369

Table 1. Comparison of aggregate signature schemes. Signatures are by l signers; k is the output length of a collision resistant hash function; "R.O." denotes if the security proof uses random oracles. Neven's scheme supports message recovery, which can reduce the effective signature overhead.

Scheme	R.O.	Sequential	Key model	Sig. size	Key size	Verification	Signing
BGLS	YES	NO	Chosen	160 bits	1920 bits	$l+1$ pair.	1 exp.
LMRS-1	YES	YES	Chosen	1024 bits	2048 bits	$2l$ exp.	verify + 1 exp.
LMRS-2	YES	YES	Registered	1024 bits	1024 bits	$4l$ mult.	verify + 1 exp.
Neven	YES	YES	Chosen	1184 bits	1024 bits	$2l$ mult.	verify + 1 exp.
Ours	NO	YES	Registered	320 bits	311040 bits	2 pair., $lk/2$ mult.	verify + 1 exp.

Credits: Material taken from the Journal of Cryptology, Springer

- The authors instantiated their pairing-based schemes using a Barreto-Naehrig (BN) curve with a 160−bit point representation. As a consequence, their schemes enjoy at most 80 bits of security

A recent example of a practical pairing-based protocol

Sequential Aggregate Signatures, Multisignatures, and Verifiably Encrypted Signatures 369

Table 1. Comparison of aggregate signature schemes. Signatures are by l signers; k is the output length of a collision resistant hash function; "R.O." denotes if the security proof uses random oracles. Neven's scheme supports message recovery, which can reduce the effective signature overhead.

Scheme	R.O.	Sequential	Key model	Sig. size	Key size	Verification	Signing
BGLS	YES	NO	Chosen	160 bits	1920 bits	$l+1$ pair.	1 exp.
LMRS-1	YES	YES	Chosen	1024 bits	2048 bits	$2l$ exp.	verify + 1 exp.
LMRS-2	YES	YES	Registered	1024 bits	1024 bits	$4l$ mult.	verify + 1 exp.
Neven	YES	YES	Chosen	1184 bits	1024 bits	$2l$ mult.	verify + 1 exp.
Ours	NO	YES	Registered	320 bits	311040 bits	2 pair., $lk/2$ mult.	verify + 1 exp.

Credits: Material taken from the Journal of Cryptology, Springer

- The authors instantiated their pairing-based schemes using a Barreto-Naehrig (BN) curve with a 160−bit point representation. As a consequence, their schemes enjoy at most 80 bits of security
- Moreover, the procedures are described in the context of symmetric pairings, whereas BN curves use asymmetric pairings. hence, decisions such as whether a variable should be defined in $\mathbb{G}_1$ or in $\mathbb{G}_2$ are left open

A recent example of a practical pairing-based protocol

Sequential Aggregate Signatures, Multisignatures, and Verifiably Encrypted Signatures 369

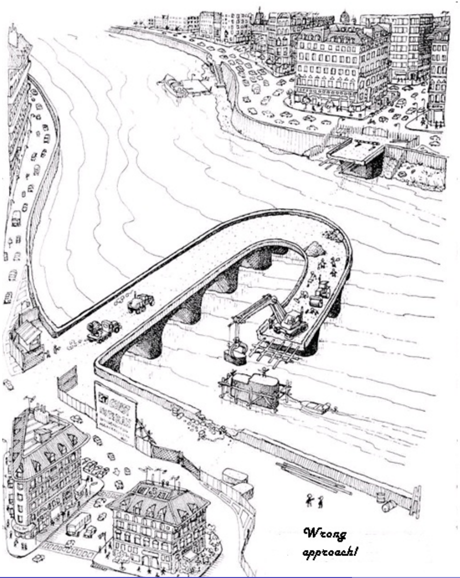
Table 1. Comparison of aggregate signature schemes. Signatures are by l signers; k is the output length of a collision resistant hash function; "R.O." denotes if the security proof uses random oracles. Neven's scheme supports message recovery, which can reduce the effective signature overhead.

Scheme	R.O.	Sequential	Key model	Sig. size	Key size	Verification	Signing
BGLS	YES	NO	Chosen	160 bits	1920 bits	$l+1$ pair.	1 exp.
LMRS-1	YES	YES	Chosen	1024 bits	2048 bits	$2l$ exp.	verify + 1 exp.
LMRS-2	YES	YES	Registered	1024 bits	1024 bits	$4l$ mult.	verify + 1 exp.
Neven	YES	YES	Chosen	1184 bits	1024 bits	$2l$ mult.	verify + 1 exp.
Ours	NO	YES	Registered	320 bits	311040 bits	2 pair., $lk/2$ mult.	verify + 1 exp.

Credits: Material taken from the Journal of Cryptology, Springer

- The authors instantiated their pairing-based schemes using a Barreto-Naehrig (BN) curve with a 160−bit point representation. As a consequence, their schemes enjoy at most 80 bits of security
- Moreover, the procedures are described in the context of symmetric pairings, whereas BN curves use asymmetric pairings. hence, decisions such as whether a variable should be defined in $\mathbb{G}_1$ or in $\mathbb{G}_2$ are left open
- The authors also pointed out that the cost of hashing to the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ is "an expensive operation". This claim is inexact at best

Pairing-based cryptography from the cryptographic implementation perspective



Some important breakthroughs on the computation of the stand-alone pairing

- 1985: Elliptic curve cryptography is independently invented by Victor Miller and Neal Koblitz

Some important breakthroughs on the computation of the stand-alone pairing

- 1985: Elliptic curve cryptography is independently invented by Victor Miller and Neal Koblitz
- 1986: Victor Miller presented an iterative algorithm that can compute the Tate pairing with linear complexity with respect to the size of the input [unpublished report until it appeared at JoC 2004]

Some important breakthroughs on the computation of the stand-alone pairing

- 1985: Elliptic curve cryptography is independently invented by Victor Miller and Neal Koblitz
- 1986: Victor Miller presented an iterative algorithm that can compute the Tate pairing with linear complexity with respect to the size of the input [unpublished report until it appeared at JoC 2004]
- 1993: MOV attack [Menezes-Okamoto-Vanstone IEEE TIT].
  [The Weil pairing was computed in 16 minutes]

Some important breakthroughs on the computation of the stand-alone pairing

- 1985: Elliptic curve cryptography is independently invented by Victor Miller and Neal Koblitz
- 1986: Victor Miller presented an iterative algorithm that can compute the Tate pairing with linear complexity with respect to the size of the input [unpublished report until it appeared at JoC 2004]
- 1993: MOV attack [Menezes-Okamoto-Vanstone IEEE TIT].
  [The Weil pairing was computed in 16 minutes]
- 2002: BKLS algorithm [Barreto-Kim-Lynn-Scott Crypto 2002]

Some important breakthroughs on the computation of the stand-alone pairing

- 1985: Elliptic curve cryptography is independently invented by Victor Miller and Neal Koblitz
- 1986: Victor Miller presented an iterative algorithm that can compute the Tate pairing with linear complexity with respect to the size of the input [unpublished report until it appeared at JoC 2004]
- 1993: MOV attack [Menezes-Okamoto-Vanstone IEEE TIT].
  [The Weil pairing was computed in 16 minutes]
- 2002: BKLS algorithm [Barreto-Kim-Lynn-Scott Crypto 2002]
- 2003: Simplifications on the Miller loop and final exponentiation computation [Duursma and Lee Asiacrypt'03]

Some important breakthroughs on the computation of the stand-alone pairing

- 1985: Elliptic curve cryptography is independently invented by Victor Miller and Neal Koblitz
- 1986: Victor Miller presented an iterative algorithm that can compute the Tate pairing with linear complexity with respect to the size of the input [unpublished report until it appeared at JoC 2004]
- 1993: MOV attack [Menezes-Okamoto-Vanstone IEEE TIT]. [The Weil pairing was computed in 16 minutes]
- 2002: BKLS algorithm [Barreto-Kim-Lynn-Scott Crypto 2002]
- 2003: Simplifications on the Miller loop and final exponentiation computation [Duursma and Lee Asiacrypt'03]
- 2006: η pairing [Hess-Smart-Vercauteren IEEE TIT]

Some important breakthroughs on the computation of the stand-alone pairing

- 1985: Elliptic curve cryptography is independently invented by Victor Miller and Neal Koblitz
- 1986: Victor Miller presented an iterative algorithm that can compute the Tate pairing with linear complexity with respect to the size of the input [unpublished report until it appeared at JoC 2004]
- 1993: MOV attack [Menezes-Okamoto-Vanstone IEEE TIT].
  [The Weil pairing was computed in 16 minutes]
- 2002: BKLS algorithm [Barreto-Kim-Lynn-Scott Crypto 2002]
- 2003: Simplifications on the Miller loop and final exponentiation computation [Duursma and Lee Asiacrypt'03]
- 2006: η pairing [Hess-Smart-Vercauteren IEEE TIT]
- 2010: optimal pairings [Vercauteren IEEE TIT]

Pairing-based cryptography from the cryptographic implementation perspective

- A lot of effort has been devoted to optimize the complexity of the stand-alone pairing

Pairing-based cryptography from the cryptographic implementation perspective

- A lot of effort has been devoted to optimize the complexity of the stand-alone pairing
- As a consequence, now a single pairing at the 128-bit security level can be computed in a fraction of millisecond both, in software and in hardware implementations

Pairing-based cryptography from the cryptographic implementation perspective

- A lot of effort has been devoted to optimize the complexity of the stand-alone pairing
- As a consequence, now a single pairing at the 128-bit security level can be computed in a fraction of millisecond both, in software and in hardware implementations
- Nevertheless, little effort has been devoted to the problem of computing product of pairings and/or multi-pairing calculations efficiently

Pairing-based cryptography from the cryptographic implementation perspective

- A lot of effort has been devoted to optimize the complexity of the stand-alone pairing
- As a consequence, now a single pairing at the 128-bit security level can be computed in a fraction of millisecond both, in software and in hardware implementations
- Nevertheless, little effort has been devoted to the problem of computing product of pairings and/or multi-pairing calculations efficiently
- Several crucial building blocks [notably the map-to-point hash function] have not been subject of a careful C level or assembly implementation

Pairing-based cryptography from the cryptographic implementation perspective

- A lot of effort has been devoted to optimize the complexity of the stand-alone pairing
- As a consequence, now a single pairing at the 128-bit security level can be computed in a fraction of millisecond both, in software and in hardware implementations
- Nevertheless, little effort has been devoted to the problem of computing product of pairings and/or multi-pairing calculations efficiently
- Several crucial building blocks [notably the map-to-point hash function] have not been subject of a careful C level or assembly implementation
- In spite of some efforts, as of today there are no good benchmarks of the relative computational costs associated to the most important building blocks of pairing-based protocols

Previous work

1. [Pairing 2013]: Chuengsatiansup et al. “PandA: Pairings and Arithmetic”
2. [ACNS 2013]: A. Guillevic. “Comparing the pairing efficiency over composite-Order and prime-order elliptic curves”
3. [ACNS 2013]: Sánchez-Ramírez and RH. “NEON implementation of an attribute-based encryption scheme”
4. [eprint 2012]: M. Scott. “Replacing username/password with software-only two-factor authentication”
5. [IMA 2011]: M. Scott. “On the efficient implementation of pairing-based protocols”
6. [Comp.Comm 2011]: Oliveira et al. “TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks”
7. [DCC 2010]: Chatterjee et al. “Comparing two pairing-based aggregate signature schemes”

Cryptographic libraries available

- Charm, a framework for rapidly prototyping cryptosystems
- RELIC is an Efficient Library for Cryptography
- MIRACL Cryptographic SDK
- PBC, The Pairing-Based Cryptography Library

A recent example of a nonconcurrence

- In Pairing 2012 there was a session named: "Implementations in Hardware and Software", it included,
  - 2 papers about the hardware implementation of a 128-bit stand-alone pairing plus,

A recent example of a nonconcurrence

- In Pairing 2012 there was a session named: "Implementations in Hardware and Software", it included,
  - 2 papers about the hardware implementation of a 128-bit stand-alone pairing plus,
  - 1 paper about the software implementation of pairings at the 192-bit security level.

A recent example of a nonconcurrence

- In Pairing 2012 there was a session named: "Implementations in Hardware and Software", it included,
  - 2 papers about the hardware implementation of a 128-bit stand-alone pairing plus,
  - 1 paper about the software implementation of pairings at the 192-bit security level.

  In spite of the highly involved implementation, none of the papers considered the option of computing a whole pairing-based protocol

A recent example of a nonconcurrence

- In Pairing 2012 there was a session named: "Implementations in Hardware and Software", it included,
  - 2 papers about the hardware implementation of a 128-bit stand-alone pairing plus,
  - 1 paper about the software implementation of pairings at the 192-bit security level.

  In spite of the highly involved implementation, none of the papers considered the option of computing a whole pairing-based protocol
- Also, In Pairing 2012 there was a session named: "Signature Schemes and Applications" that included three nice papers about different signature schemes.

A recent example of a nonconcurrence

- In Pairing 2012 there was a session named: "Implementations in Hardware and Software", it included,
  - 2 papers about the hardware implementation of a 128-bit stand-alone pairing plus,
  - 1 paper about the software implementation of pairings at the 192-bit security level.

  In spite of the highly involved implementation, none of the papers considered the option of computing a whole pairing-based protocol
- Also, In Pairing 2012 there was a session named: "Signature Schemes and Applications" that included three nice papers about different signature schemes.

  Once again, none of these three papers had a section dealing with implementation considerations

A recent example of a nonconcurrence

- A plausible conclusion is that everybody is having fun in his/her own way, which is great... but

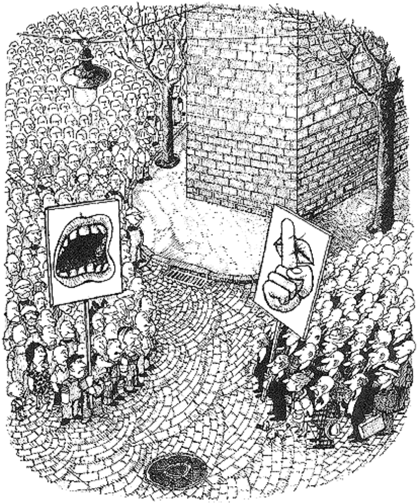
A recent example of a nonconcurrence

- A plausible conclusion is that everybody is having fun in his/her own way, which is great... but
- Wouldn't it be also nice to have soon in Pairing a session named something like:
  "Efficient implementation of pairing-based protocols"
  with contributed papers produced by a mixed team of protocol designers and cryptographic implementors?

A recent example of a nonconcurrence

- A plausible conclusion is that everybody is having fun in his/her own way, which is great... but
- Wouldn't it be also nice to have soon in Pairing a session named something like:
  "Efficient implementation of pairing-based protocols"
  with contributed papers produced by a mixed team of protocol designers and cryptographic implementors?
- The purpose of this talk is to present some steps forward in that direction

Implementing popular cryptographic building blocks used in pairing-based protocols



borrowed from Quino.

Barreto-Naehrig curves

Barreto-Naehrig (BN) elliptic curves are a family of elliptic curves with embedding degree $k = 12$ defined by the equation

$$E/\mathbb{F}_p : y^2 = x^3 + b, b \neq 0,$$

where the prime p, the group order $r = \#E(\mathbb{F}_p)$, and the trace of Frobenius t are parametrized as,

$$\begin{aligned} p(u) &= 36u^4 + 36u^3 + 24u^2 + 6u + 1; \\ r(u) &= 36u^4 + 36u^3 + 18u^2 + 6u + 1; \\ t(u) &= 6u^2 + 1, \end{aligned}$$

where $u \in \mathbb{Z}$

Barreto-Naehrig curves

Barreto-Naehrig (BN) elliptic curves are a family of elliptic curves with embedding degree $k = 12$ defined by the equation

$$E/\mathbb{F}_p : y^2 = x^3 + b, b \neq 0,$$

where the prime p, the group order $r = \#E(\mathbb{F}_p)$, and the trace of Frobenius t are parametrized as,

$$\begin{aligned} p(u) &= 36u^4 + 36u^3 + 24u^2 + 6u + 1; \\ r(u) &= 36u^4 + 36u^3 + 18u^2 + 6u + 1; \\ t(u) &= 6u^2 + 1, \end{aligned}$$

where $u \in \mathbb{Z}$
BN curves admit a sextic twist curve, defined as $\tilde{E}(\mathbb{F}_{p^2}) : Y^2 = X^3 + b/\xi$, where $\xi \in \mathbb{F}_{p^2}$ is neither a square nor a cube in $\mathbb{F}_{p^2}$.

Hashing to $\mathbb{G}_1$

The Map-to-point hash function H_1 to the group $\mathbb{G}_1$ is defined as,

$$H_1 : \{0,1\}^* \to \mathbb{G}_1^*.$$

Hashing to $\mathbb{G}_1$

The Map-to-point hash function H_1 to the group $\mathbb{G}_1$ is defined as,

$$H_1 : \{0,1\}^* \to \mathbb{G}_1^*.$$

In the case of BN curves one can use the following procedure:

1. Use a standard cryptographic hash function to map an arbitrary string to a field element $\tau \in \mathbb{F}_p$

Hashing to $\mathbb{G}_1$

The Map-to-point hash function H_1 to the group $\mathbb{G}_1$ is defined as,

$$H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*.$$

In the case of BN curves one can use the following procedure:

1. Use a standard cryptographic hash function to map an arbitrary string to a field element $\tau \in \mathbb{F}_p$
2. Map the field element τ to a point in the group $\mathbb{G}_1$ using a deterministic BN hash procedure [Fouque and Tibouchi, Latincrypt 2012]

Hashing to $\mathbb{G}_2$

The Map-to-point hash function H_1 to the group $\mathbb{G}_2$ is defined as,

$$H_1 : \{0,1\}^* \rightarrow \mathbb{G}_2^*.$$

Hashing to $\mathbb{G}_2$

The Map-to-point hash function H_1 to the group $\mathbb{G}_2$ is defined as,

$$H_1 : \{0,1\}^* \rightarrow \mathbb{G}_2^*.$$

In the case of BN curves one can use the following procedure:

1. Use a standard cryptographic hash function to map an arbitrary string to a field element $\tau \in \mathbb{F}_{p^2}$

Hashing to $\mathbb{G}_2$

The Map-to-point hash function H_1 to the group $\mathbb{G}_2$ is defined as,

$$H_1 : \{0,1\}^* \rightarrow \mathbb{G}_2^*.$$

In the case of BN curves one can use the following procedure:

1. Use a standard cryptographic hash function to map an arbitrary string to a field element $\tau \in \mathbb{F}_{p^2}$
2. Map the field element τ to a point in the group $\mathbb{G}_2$ using an adapted version of the BN hash procedure by [Fouque and Tibouchi, Latincrypt 2012]

Hashing to $\mathbb{G}_2$

The Map-to-point hash function H_1 to the group $\mathbb{G}_2$ is defined as,

$$H_1 : \{0, 1\}^* \to \mathbb{G}_2^*.$$

In the case of BN curves one can use the following procedure:

1. Use a standard cryptographic hash function to map an arbitrary string to a field element $\tau \in \mathbb{F}_{p^2}$
2. Map the field element τ to a point in the group $\mathbb{G}_2$ using an adapted version of the BN hash procedure by [Fouque and Tibouchi, Latincrypt 2012]
3. Compute the scalar multiplication cQ, where the cofactor c is given as, $c = \#\tilde{E}(\mathbb{F}_{p^2})/r$. This scalar multiplication can be greatly accelerated using the Frobenius endomorphism plus lattice reduction techniques [Fuentes-Castañeda, Knapp and RH, SAC 2011]

Scalar multiplication in $\mathbb{G}_1$

Problem:

Given the point $P \in \mathbb{G}_1$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $R = nP$

Scalar multiplication in $\mathbb{G}_1$

Problem:

Given the point $P \in \mathbb{G}_1$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $R = nP$

- BN curves admit a two-dimensional GLV method that yields significant computational savings for this operation

Scalar multiplication in $\mathbb{G}_1$

Problem:

Given the point $P \in \mathbb{G}_1$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $R = nP$

- BN curves admit a two-dimensional GLV method that yields significant computational savings for this operation
- We distinguish two cases depending if the point P is known in advance or not

Scalar multiplication in $\mathbb{G}_1$

Problem:
Given the point $P \in \mathbb{G}_1$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $R = nP$

- BN curves admit a two-dimensional GLV method that yields significant computational savings for this operation
- We distinguish two cases depending if the point P is known in advance or not
- computing a scalar multiplication with a fixed point is roughly 3.2 times faster than computing it with a variable one

Scalar multiplication in $\mathbb{G}_1$

Problem:

Given the point $P \in \mathbb{G}_1$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $R = nP$

- BN curves admit a two-dimensional GLV method that yields significant computational savings for this operation
- We distinguish two cases depending if the point P is known in advance or not
- computing a scalar multiplication with a fixed point is roughly 3.2 times faster than computing it with a variable one
- However, fixed point scalar multiplications use a pre-computed table of 128 points whereas variable point scalar multiplications only require two precomputed points

Scalar multiplication in $\mathbb{G}_2$, exponentiation in $\mathbb{G}_T$

Problems:
Given the point $Q \in \mathbb{G}_2$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $S = nQ$

Scalar multiplication in $\mathbb{G}_2$, exponentiation in $\mathbb{G}_T$

Problems:

Given the point $Q \in \mathbb{G}_2$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $S = nQ$

Given a field element $f \in \mathbb{G}_T$ and the exponent $n \in \mathbb{F}_r$ one wants to compute the powering $g = f^n$

Scalar multiplication in $\mathbb{G}_2$, exponentiation in $\mathbb{G}_T$

Problems:

Given the point $Q \in \mathbb{G}_2$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $S = nQ$

Given a field element $f \in \mathbb{G}_T$ and the exponent $n \in \mathbb{F}_r$ one wants to compute the powering $g = f^n$

- BN curves admit a four-dimensional GS method that yields significant computational savings for these two operations

Scalar multiplication in $\mathbb{G}_2$, exponentiation in $\mathbb{G}_T$

Problems:

Given the point $Q \in \mathbb{G}_2$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $S = nQ$

Given a field element $f \in \mathbb{G}_T$ and the exponent $n \in \mathbb{F}_r$ one wants to compute the powering $g = f^n$

- BN curves admit a four-dimensional GS method that yields significant computational savings for these two operations
- In these groups, computing a scalar multiplication/exponentiation with a fixed point is roughly 2.2 times faster than computing it with a variable one

Scalar multiplication in $\mathbb{G}_2$, exponentiation in $\mathbb{G}_T$

Problems:

Given the point $Q \in \mathbb{G}_2$ and the scalar $n \in \mathbb{F}_r$ one wants to compute the multiple $S = nQ$

Given a field element $f \in \mathbb{G}_T$ and the exponent $n \in \mathbb{F}_r$ one wants to compute the powering $g = f^n$

- BN curves admit a four-dimensional GS method that yields significant computational savings for these two operations
- In these groups, computing a scalar multiplication/exponentiation with a fixed point is roughly 2.2 times faster than computing it with a variable one
- However, fixed point scalar multiplications/exponentiations use a pre-computed table of 128 points whereas variable point scalar multiplications/exponentiations only require two precomputed points

Single pairing computation

Problem:

Given the points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, compute the pairing $\hat{e}(Q, P)$.

Single pairing computation

Problem:

Given the points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, compute the pairing $\hat{e}(Q, P)$.

- We used the optimal ate pairing over BN curves as it was formulated in [Aranha *et al.* Eurocrypt'2011].

Single pairing computation

Problem:

Given the points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, compute the pairing $\hat{e}(Q, P)$.

- We used the optimal ate pairing over BN curves as it was formulated in [Aranha *et al.* Eurocrypt'2011].
- The arithmetic operations are performed using field towering techniques, where the extension field $\mathbb{F}_{p^{12}}$ is represented as the tower,

$$\mathbb{F}_p \subset \mathbb{F}_{p^2} \subset \mathbb{F}_{p^6} \subset \mathbb{F}_{p^{12}}.$$

Single pairing computation

Problem:

Given the points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, compute the pairing $\hat{e}(Q, P)$.

- We used the optimal ate pairing over BN curves as it was formulated in [Aranha *et al.* Eurocrypt'2011].
- The arithmetic operations are performed using field towering techniques, where the extension field $\mathbb{F}_{p^{12}}$ is represented as the tower,

$$\mathbb{F}_p \subset \mathbb{F}_{p^2} \subset \mathbb{F}_{p^6} \subset \mathbb{F}_{p^{12}}.$$

- The two most dominant arithmetic operations are,
  - the integer multiplication of two 256-bit integers (producing a 512-bit integer), denoted as m_E and,
  - The Montgomery reduction from a 512-bit integer to a 256-bit integer, denoted as r_E.

Single pairing computation

Problem:

Given the points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, compute the pairing $\hat{e}(Q, P)$.

- We used the optimal ate pairing over BN curves as it was formulated in [Aranha *et al.* Eurocrypt'2011].
- The arithmetic operations are performed using field towering techniques, where the extension field $\mathbb{F}_{p^{12}}$ is represented as the tower,

$$\mathbb{F}_p \subset \mathbb{F}_{p^2} \subset \mathbb{F}_{p^6} \subset \mathbb{F}_{p^{12}}.$$

- The two most dominant arithmetic operations are,
  - the integer multiplication of two 256-bit integers (producing a 512-bit integer), denoted as m_E and,
  - The Montgomery reduction from a 512-bit integer to a 256-bit integer, denoted as r_E.
- We chose the BN parameter as $u = -2^{62} - 2^{55} - 1$ that yields a 254-bit prime p.

Single pairing computation

Problem:

Given the points $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, compute the pairing $\hat{e}(Q, P)$.

- We used the optimal ate pairing over BN curves as it was formulated in [Aranha *et al.* Eurocrypt'2011].
- The arithmetic operations are performed using field towering techniques, where the extension field $\mathbb{F}_{p^{12}}$ is represented as the tower,

$$\mathbb{F}_p \subset \mathbb{F}_{p^2} \subset \mathbb{F}_{p^6} \subset \mathbb{F}_{p^{12}}.$$

- The two most dominant arithmetic operations are,
  - the integer multiplication of two 256-bit integers (producing a 512-bit integer), denoted as m_E and,
  - The Montgomery reduction from a 512-bit integer to a 256-bit integer, denoted as r_E.
- We chose the BN parameter as $u = -2^{62} - 2^{55} - 1$ that yields a 254-bit prime p.
- If the point Q is known in advance, the pairing can be computed $\approx 15\%$ faster using precomputation

Multipairing computation

- Many protocols require the computation of product of pairings

Multipairing computation

- Many protocols require the computation of product of pairings
- If all the pairings share a common input point, then use,

$$\prod_{i=0}^{n-1} e(Q, P_i) = e(Q, \sum_{i=0}^{n-1} P_i),$$

Multipairing computation

- Many protocols require the computation of product of pairings
- If all the pairings share a common input point, then use,

$$\prod_{i=0}^{n-1} e(Q, P_i) = e(Q, \sum_{i=0}^{n-1} P_i),$$

- Otherwise, the pairing can still be accelerated by using strategies analogous to multi-exponentiation computations
  [all the point doubling computations in the Miller loop can be shared using a single accumulator]

Popular cryptographic building blocks for pairing-based protocols

Operation	Operation count	10^3 Clock cycles	op/$\mathbb{G}_1$_UK_mul
single pairing (unknown point)	$10,312 m_E + 4,954 r_E$	1,162	5.95
single pairing (known point)	$8,738 m_E + 3,792 r_E$	980	5.03
1 more pairing (unknown point)	$4,604 m_E + 2,301 r_E$	483	2.48
1 more pairing (known point)	$3,011 m_E + 1,125 r_E$	280	1.44
known sc. mult. in $\mathbb{G}_1$, $w = 8$	$\approx 576m$	61	0.31
unknown sc. mult. in $\mathbb{G}_1$, $w = 3$	$\approx 1,654m$	195	1.00
known sc. mult. in $\mathbb{G}_2$, $w = 8$	$\approx 1,472m$	161	0.83
unknown sc. mult. in $\mathbb{G}_2$, $w = 3$	$\approx 3,036m$	354	1.82
known exp. in $\mathbb{G}_T$, $w = 8$	$\approx 2,496m$	260	1.33
unknown exp. in $\mathbb{G}_T$, $w = 3$	$\approx 5,070m$	557	2.86
Map-To-Point $\mathbb{G}_1$	$\approx 750m$	72	0.37
Map-To-Point $\mathbb{G}_2$	$\approx 2,760m$	262	1.34

Table: m_E, r_E and m denote 256-bit integer multiplication, 512-bit Montgomery reduction and field multiplication over $\mathbb{F}_p$, respectively. All the scalar multiplications/exponentiations process the scalar/exponent using a window size w as indicated.

Timings measured on an Intel Core i7-4770 with the micro-architecture Haswell running at 3.4GHz

Popular cryptographic building blocks for pairing-based protocols

Operation	Operation count	10^3 Clock cycles	op/$\mathbb{G}_1$_UK_mul
single pairing (unknown point)	$10,312 m_E + 4,954 r_E$	1,162	5.95
single pairing (known point)	$8,738 m_E + 3,792 r_E$	980	5.03
1 more pairing (unknown point)	$4,604 m_E + 2,301 r_E$	483	2.48
1 more pairing (known point)	$3,011 m_E + 1,125 r_E$	280	1.44
known sc. mult. in $\mathbb{G}_1$, $w = 8$	$\approx 576m$	61	0.31
unknown sc. mult. in $\mathbb{G}_1$, $w = 3$	$\approx 1,654m$	195	1.00
known sc. mult. in $\mathbb{G}_2$, $w = 8$	$\approx 1,472m$	161	0.83
unknown sc. mult. in $\mathbb{G}_2$, $w = 3$	$\approx 3,036m$	354	1.82
known exp. in $\mathbb{G}_T$, $w = 8$	$\approx 2,496m$	260	1.33
unknown exp. in $\mathbb{G}_T$, $w = 3$	$\approx 5,070m$	557	2.86
Map-To-Point $\mathbb{G}_1$	$\approx 750m$	72	0.37
Map-To-Point $\mathbb{G}_2$	$\approx 2,760m$	262	1.34

Table: m_E, r_E and m denote 256-bit integer multiplication, 512-bit Montgomery reduction and field multiplication over $\mathbb{F}_p$, respectively. All the scalar multiplications/exponentiations process the scalar/exponent using a window size w as indicated.

Timings measured on an Intel Core i7-4770 with the micro-architecture Haswell running at 3.4GHz

Popular cryptographic building blocks for pairing-based protocols

Operation	Operation count	10^3 Clock cycles	op/$\mathbb{G}_1$_UK_mul
single pairing (unknown point)	$10,312m_E + 4,954r_E$	1,162	5.95
single pairing (known point)	$8,738m_E + 3,792r_E$	980	5.03
1 more pairing (unknown point)	$4,604m_E + 2,301r_E$	483	2.48
1 more pairing (known point)	$3,011m_E + 1,125r_E$	280	1.44
known sc. mult. in $\mathbb{G}_1$, $w = 8$	$\approx 576m$	61	0.31
unknown sc. mult. in $\mathbb{G}_1$, $w = 3$	$\approx 1,654m$	195	1.00
known sc. mult. in $\mathbb{G}_2$, $w = 8$	$\approx 1,472m$	161	0.83
unknown sc. mult. in $\mathbb{G}_2$, $w = 3$	$\approx 3,036m$	354	1.82
known exp. in $\mathbb{G}_T$, $w = 8$	$\approx 2,496m$	260	1.33
unknown exp. in $\mathbb{G}_T$, $w = 3$	$\approx 5,070m$	557	2.86
Map-To-Point $\mathbb{G}_1$	$\approx 750m$	72	0.37
Map-To-Point $\mathbb{G}_2$	$\approx 2,760m$	262	1.34

Table: m_E, r_E and m denote 256-bit integer multiplication, 512-bit Montgomery reduction and field multiplication over $\mathbb{F}_p$, respectively. All the scalar multiplications/exponentiations process the scalar/exponent using a window size w as indicated.

Timings measured on an Intel Core i7-4770 with the micro-architecture Haswell running at 3.4GHz

Popular cryptographic building blocks for pairing-based protocols

Operation	Operation count	10^3 Clock cycles	op/$\mathbb{G}_1$_UK_mul
single pairing (unknown point)	$10,312 m_E + 4,954 r_E$	1,162	5.95
single pairing (known point)	$8,738 m_E + 3,792 r_E$	980	5.03
1 more pairing (unknown point)	$4,604 m_E + 2,301 r_E$	483	2.48
1 more pairing (known point)	$3,011 m_E + 1,125 r_E$	280	1.44
known sc. mult. in $\mathbb{G}_1$, $w = 8$	$\approx 576m$	61	0.31
unknown sc. mult. in $\mathbb{G}_1$, $w = 3$	$\approx 1,654m$	195	1.00
known sc. mult. in $\mathbb{G}_2$, $w = 8$	$\approx 1,472m$	161	0.83
unknown sc. mult. in $\mathbb{G}_2$, $w = 3$	$\approx 3,036m$	354	1.82
known exp. in $\mathbb{G}_T$, $w = 8$	$\approx 2,496m$	260	1.33
unknown exp. in $\mathbb{G}_T$, $w = 3$	$\approx 5,070m$	557	2.86
Map-To-Point $\mathbb{G}_1$	$\approx 750m$	72	0.37
Map-To-Point $\mathbb{G}_2$	$\approx 2,760m$	262	1.34

Table: m_E, r_E and m denote 256-bit integer multiplication, 512-bit Montgomery reduction and field multiplication over $\mathbb{F}_p$, respectively. All the scalar multiplications/exponentiations process the scalar/exponent using a window size w as indicated.

Timings measured on an Intel Core i7-4770 with the micro-architecture Haswell running at 3.4GHz

Popular cryptographic building blocks for pairing-based protocols

Operation	Operation count	10^3 Clock cycles	op/$\mathbb{G}_1$_UK_mul
single pairing (unknown point)	$10,312 m_E + 4,954 r_E$	1,162	5.95
single pairing (known point)	$8,738 m_E + 3,792 r_E$	980	5.03
1 more pairing (unknown point)	$4,604 m_E + 2,301 r_E$	483	2.48
1 more pairing (known point)	$3,011 m_E + 1,125 r_E$	280	1.44
known sc. mult. in $\mathbb{G}_1$, $w = 8$	$\approx 576 m$	61	0.31
unknown sc. mult. in $\mathbb{G}_1$, $w = 3$	$\approx 1,654 m$	195	1.00
known sc. mult. in $\mathbb{G}_2$, $w = 8$	$\approx 1,472 m$	161	0.83
unknown sc. mult. in $\mathbb{G}_2$, $w = 3$	$\approx 3,036 m$	354	1.82
known exp. in $\mathbb{G}_T$, $w = 8$	$\approx 2,496 m$	260	1.33
unknown exp. in $\mathbb{G}_T$, $w = 3$	$\approx 5,070 m$	557	2.86
Map-To-Point $\mathbb{G}_1$	$\approx 750 m$	72	0.37
Map-To-Point $\mathbb{G}_2$	$\approx 2,760 m$	262	1.34

Table: m_E, r_E and m denote 256-bit integer multiplication, 512-bit Montgomery reduction and field multiplication over $\mathbb{F}_p$, respectively. All the scalar multiplications/exponentiations process the scalar/exponent using a window size w as indicated.

Timings measured on an Intel Core i7-4770 with the micro-architecture Haswell running at 3.4GHz

Popular cryptographic building blocks for pairing-based protocols

Operation	Operation count	10^3 Clock cycles	op/$\mathbb{G}_1$_UK_mul
single pairing (unknown point)	$10,312m_E + 4,954r_E$	1,162	5.95
single pairing (known point)	$8,738m_E + 3,792r_E$	980	5.03
1 more pairing (unknown point)	$4,604m_E + 2,301r_E$	483	2.48
1 more pairing (known point)	$3,011m_E + 1,125r_E$	280	1.44
known sc. mult. in $\mathbb{G}_1$, $w = 8$	$\approx 576m$	61	0.31
unknown sc. mult. in $\mathbb{G}_1$, $w = 3$	$\approx 1,654m$	195	1.00
known sc. mult. in $\mathbb{G}_2$, $w = 8$	$\approx 1,472m$	161	0.83
unknown sc. mult. in $\mathbb{G}_2$, $w = 3$	$\approx 3,036m$	354	1.82
known exp. in $\mathbb{G}_T$, $w = 8$	$\approx 2,496m$	260	1.33
unknown exp. in $\mathbb{G}_T$, $w = 3$	$\approx 5,070m$	557	2.86
Map-To-Point $\mathbb{G}_1$	$\approx 750m$	72	0.37
Map-To-Point $\mathbb{G}_2$	$\approx 2,760m$	262	1.34

Table: m_E, r_E and m denote 256-bit integer multiplication, 512-bit Montgomery reduction and field multiplication over $\mathbb{F}_p$, respectively. All the scalar multiplications/exponentiations process the scalar/exponent using a window size w as indicated.

Timings measured on an Intel Core i7-4770 with the micro-architecture Haswell running at 3.4GHz

Popular cryptographic building blocks for pairing-based protocols

Operation	Operation count	10^3 Clock cycles	op/$\mathbb{G}_1$_UK_mul
single pairing (unknown point)	$10,312 m_E + 4,954 r_E$	1,162	5.95
single pairing (known point)	$8,738 m_E + 3,792 r_E$	980	5.03
1 more pairing (unknown point)	$4,604 m_E + 2,301 r_E$	483	2.48
1 more pairing (known point)	$3,011 m_E + 1,125 r_E$	280	1.44
known sc. mult. in $\mathbb{G}_1$, $w = 8$	$\approx 576m$	61	0.31
unknown sc. mult. in $\mathbb{G}_1$, $w = 3$	$\approx 1,654m$	195	1.00
known sc. mult. in $\mathbb{G}_2$, $w = 8$	$\approx 1,472m$	161	0.83
unknown sc. mult. in $\mathbb{G}_2$, $w = 3$	$\approx 3,036m$	354	1.82
known exp. in $\mathbb{G}_T$, $w = 8$	$\approx 2,496m$	260	1.33
unknown exp. in $\mathbb{G}_T$, $w = 3$	$\approx 5,070m$	557	2.86
Map-To-Point $\mathbb{G}_1$	$\approx 750m$	72	0.37
Map-To-Point $\mathbb{G}_2$	$\approx 2,760m$	262	1.34

Table: m_E, r_E and m denote 256-bit integer multiplication, 512-bit Montgomery reduction and field multiplication over $\mathbb{F}_p$, respectively. All the scalar multiplications/exponentiations process the scalar/exponent using a window size w as indicated.

Timings measured on an Intel Core i7-4770 with the micro-architecture Haswell running at 3.4GHz

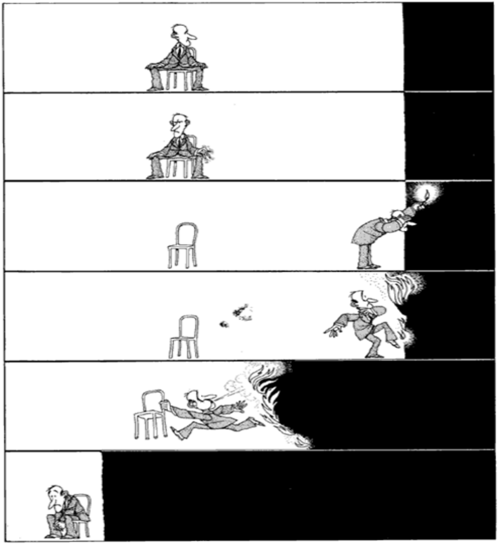
Popular cryptographic building blocks for pairing-based protocols

Operation	10^3 Clock cycles	op/$\mathbb{G}_1$_UK_mul
single pairing	$5,838$	7.10
known sc. mult. in $\mathbb{G}_1$, $w = 8$	251	0.31
unknown sc. mult. in $\mathbb{G}_1$, $w = 3$	822	1.00
known sc. mult. in $\mathbb{G}_2$, $w = 8$	636	0.77
unknown sc. mult. in $\mathbb{G}_2$, $w = 3$	$1,571$	1.91
known exp. in $\mathbb{G}_T$, $w = 8$	$1,121$	1.36
unknown exp. in $\mathbb{G}_T$, $w = 3$	$2,522$	3.06

Table: All the scalar multiplications/exponentiations process the scalar/exponent using a window size w as indicated.

Timings measured on an Exynos 5 Cortex-A15 running at 1.7GHz using NEON as reported in [Sánchez-Ramírez and RH ACNS 2013]

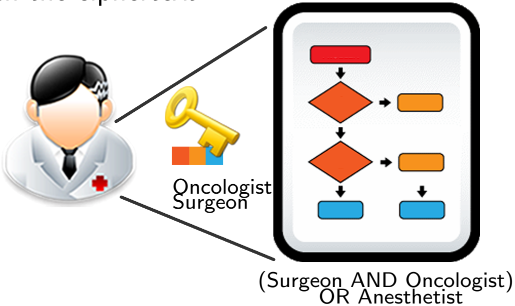
Case Study: Attribute-based encryption



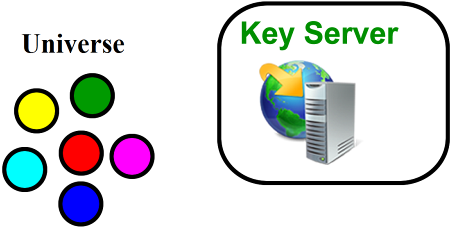
borrowed from Quino.

Attribute-based encryption overview

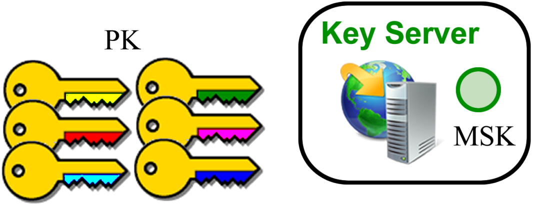
- In 2004, Sahai and Waters introduced attribute-based encryption (ABE) as a new method for encrypted access control
- In this scheme both, the user's private key and the ciphertext are associated with a set of attributes or with an access policy defined by a set of attributes
- A user can decrypt the ciphertext if her private key satisfies the access policy associated with the ciphertext or covers the set of attributes related with the ciphertext.



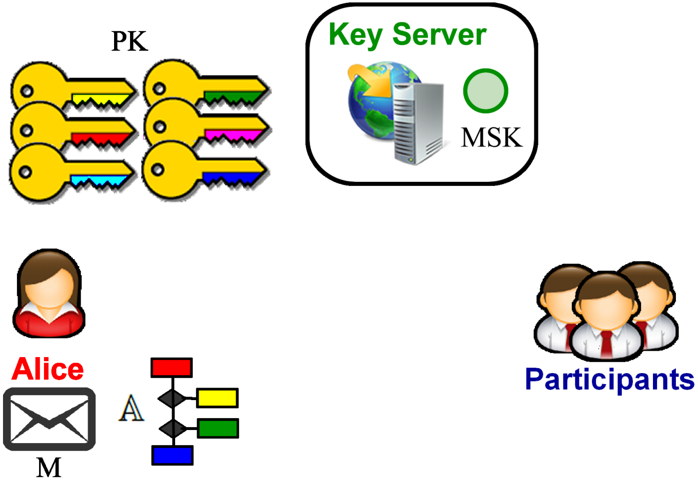
Attribute-based encryption overview



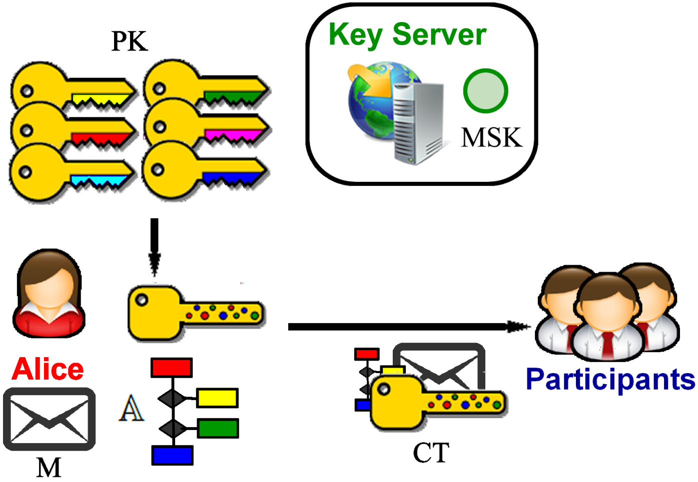
Attribute-based encryption overview



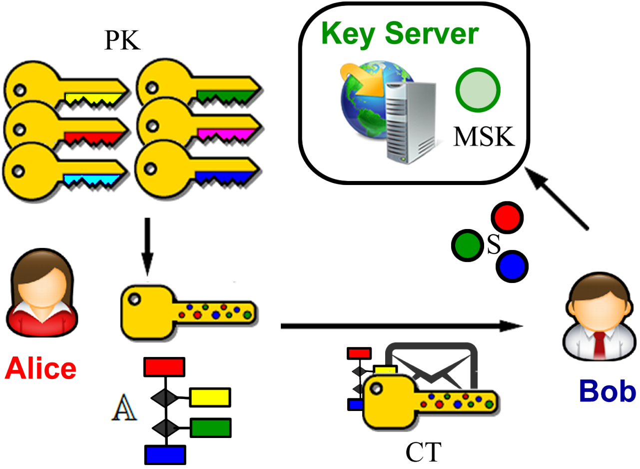
Attribute-based encryption overview



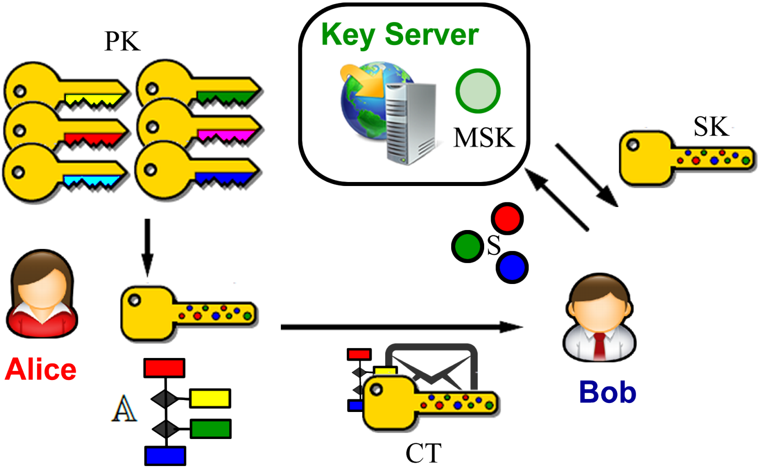
Attribute-based encryption overview



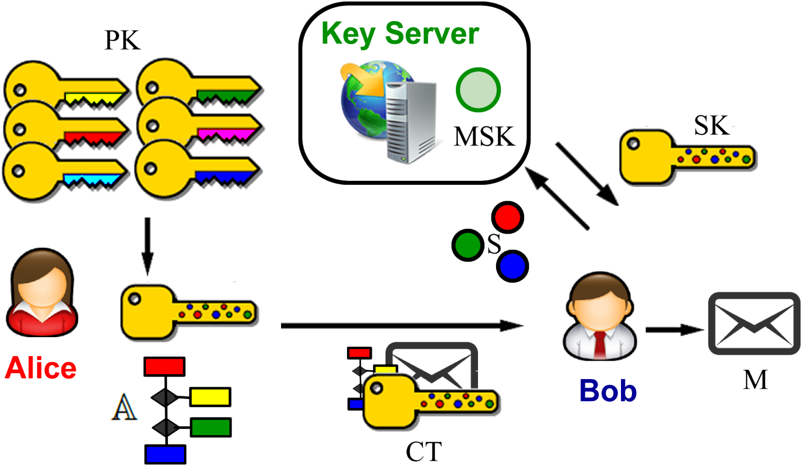
Attribute-based encryption overview



Attribute-based encryption overview



Attribute-based encryption overview



Access Policy

- The access policy $\mathbb{A}$ is initially specified as a boolean formula over a subset of attributes. Let us assume that the number of distinct attributes in that boolean formula is u
- The boolean formula describing the access policy is converted into a Linear Secret-Sharing Scheme (LSSS) matrix $\mathcal{S} \in \mathbb{F}_r$ of size $u \times t$, along with a function ρ that associates rows of $\mathcal{S}$ to attributes in $\mathcal{H}$. Here t is the number of shares to be produced.

$$\mathcal{S} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1t} \\ a_{21} & a_{22} & \dots & a_{2t} \\ \vdots & \vdots & \ddots & \vdots \\ a_{u1} & a_{u2} & \dots & a_{ut} \end{bmatrix} \xrightarrow[\rho(i)]{} \begin{bmatrix} \text{Attribute}_1 \\ \text{Attribute}_2 \\ \vdots \\ \text{Attribute}_u \end{bmatrix}$$

Sharing/recovering a secret in LSSS

- Sharing a secret. Let us consider the column vector $\bar{u} = (s, y_2, \ldots, y_t)$, then $\bar{\lambda} = \mathcal{S}\bar{u}$ is the vector of t shares of the secret s, where λ_i belongs to the attribute $\rho(i)$.

Sharing/recovering a secret in LSSS

- Sharing a secret. Let us consider the column vector $\bar{u} = (s, y_2, \ldots, y_t)$, then $\bar{\lambda} = \mathcal{S}\bar{u}$ is the vector of t shares of the secret s, where λ_i belongs to the attribute $\rho(i)$.
- Recovering a secret. An LSSS matrix $\mathcal{S}$, is reduced to a square matrix by removing the rows and columns that are unrelated to an authorized set of attributes $\mathcal{H}$. Denote the resulting reduced $v \times v$ matrix as $\tilde{\mathcal{S}} \in \mathbb{F}_r$, with $v \leq u$. Define $\mathcal{I} \subset \{1, 2, \ldots, \ell\}$ as $\mathcal{I} = \{i : \rho(i) \in S\}$. Then, there exist constants $\{\omega_i \in \mathbb{F}_r\}_{i \in I}$ such that

$$\sum_{i \in I} \omega_i \lambda_i = s$$

.

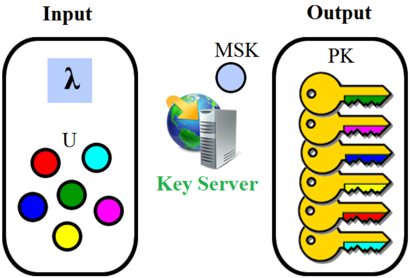
Attribute-based encryption

The four main primitives of Attribute-based encryption are,

- Setup
- Encryption
- Key Generation
- Decryption

Attribute-based encryption: setup

U

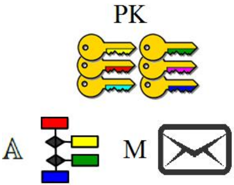


$$\mathbf{PK} = \begin{cases} P, Q, \\ \hat{e}(Q, P)^{\alpha}, \\ aP, \\ H_1, \ldots, H_U \end{cases}$$
$$\mathbf{MSK} = \alpha P.$$

$$\mathbb{G}_1, \mathbb{G}_2$$
$$P \in \mathbb{G}_1, Q \in \mathbb{G}_2$$
$$a, b \in \mathbb{F}_r$$

Attribute-based encryption: encryption

INPUT



OUTPUT

$$\mathrm{PK} = \begin{cases} P, Q, \\ \hat{e}(Q,P)^{\alpha}, \\ aP, \\ H_1, \dots, H_U \end{cases}$$

$\mathbb{A} = (\mathcal{S}, \rho)$
$\mathcal{M}$

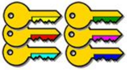
Run a LSSS with t shares (s, λ_i).
Generate $k_1, \dots, k_u \in \mathbb{Z}_r$.

$$\mathrm{CT} = \begin{cases} C = \mathcal{M} \cdot \hat{e}(Q,P)^{\alpha s}, \\ C' = sQ, \\ C_i = \lambda_i(aP) - k_i H_{\rho(i)}, \\ D_i = k_i Q \end{cases}$$

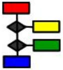
Attribute-based encryption: encryption

INPUT

PK



$\mathbb{A}$



M

Alice

OUTPUT

CT

$$\mathrm{PK} = \begin{cases} P, Q, \\ \hat{e}(Q,P)^{\alpha}, \\ aP, \\ H_1, \ldots, H_U \end{cases}$$

$\mathbb{A} = (\mathcal{S}, \rho)$

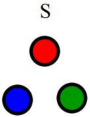
$\mathcal{M}$

Run a LSSS with t shares (s, λ_i).
Generate $k_1, \ldots, k_u \in \mathbb{Z}_r$.

$$\mathrm{CT} = \begin{cases} C = \mathcal{M} \cdot \hat{e}(Q,P)^{\alpha s}, \\ C' = sQ, \\ C_i = \lambda_i(aP) - k_i H_{\rho(i)}, \\ D_i = k_i Q \end{cases}$$

Attribute-based encryption: key generation

INPUT



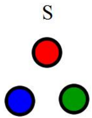
OUTPUT

$$\mathrm{PK} = \begin{cases} P, Q, \\ \hat{e}(Q, P)^{\alpha}, \\ aP, \\ H_1, \ldots, H_U \end{cases}$$

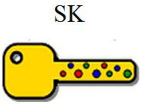
$$\mathrm{MSK} = \alpha P$$

Attribute-based encryption: key generation

INPUT



OUTPUT



$$\mathrm{PK} = \begin{cases} P, Q, \\ \hat{e}(Q, P)^{\alpha}, \\ aP, \\ H_1, \ldots, H_U \end{cases}$$

$$\mathrm{MSK} = \alpha P$$
Select $\tau \in \mathbb{Z}_r$.

$$\mathrm{SK} = \begin{cases} K = \alpha P + \tau(aP), \\ L = \tau Q, \\ \forall x \in S \;\; K_x = \tau H_x \end{cases}$$

Decryption

INPUT

CT

Bob

OUTPUT

$$\text{CT} = \begin{cases} C = \mathcal{M} \cdot e(Q, P)^{\alpha s}, \\ C' = sQ, \\ C_i = \lambda_i(aP) - k_i H_{\rho(i)}, \\ D_i = k_i Q \end{cases}$$

$$\text{SK} = \begin{cases} K = \alpha P + \tau(aP), \\ L = \tau Q, \\ \forall x \in S \;\; K_x = \tau H_x \end{cases}$$

Decryption

INPUT

CT

Bob

OUTPUT

$$\frac{\left(e(L, \sum_{i\in\mathcal{I}} \omega_i C_i)\prod_{i\in\mathcal{I}} e(D_i, \omega_i K_{\rho(i)})\right)}{e(C', K)}$$

$$= e(Q, P)^{-\alpha s}$$

$$\mathsf{CT} = \begin{cases} C = \mathcal{M} \cdot e(Q, P)^{\alpha s}, \\ C' = sQ, \\ C_i = \lambda_i(aP) - k_i H_{\rho(i)}, \\ D_i = k_i Q \end{cases}$$

$$\mathsf{SK} = \begin{cases} K = \alpha P + \tau(aP), \\ L = \tau Q, \\ \forall x \in S \;\; K_x = \tau H_x \end{cases}$$

Decryption

INPUT

CT

$$\text{CT} = \begin{cases} C = \mathcal{M} \cdot e(Q,P)^{\alpha s}, \\ C' = sQ, \\ C_i = \lambda_i(aP) - k_i H_{\rho(i)}, \\ D_i = k_i Q \end{cases}$$

$$\text{SK} = \begin{cases} K = \alpha P + \tau(aP), \\ L = \tau Q, \\ \forall x \in S \;\; K_x = \tau H_x \end{cases}$$

Bob

$$\frac{\left(e(L, \sum_{i \in \mathcal{I}} \omega_i C_i) \prod_{i \in \mathcal{I}} e(D_i, \omega_i K_{\rho(i)})\right)}{e(C', K)}$$

$$= e(Q,P)^{-\alpha s}$$

OUTPUT

M

$\mathcal{M}$

Table with the timing costs of the Attribute-based protocol

LSSS ABE Protocol	10^3 clock cycles			
	Attributes in $\mathbb{G}_1$		Attributes in $\mathbb{G}_2$	
	Six attributes	Twenty attributes	Six attributes	Twenty attributes
encryption	2,384	7,150	2,921	9,129
key generation	652	1,699	1,326	3,994
decryption ($\Delta = 1$)	4,606	12,776	3,515	9,528
overall cost	7,642	21,625	7,762	22,651
pairing cost	4,378	11,168	3,123	7,043
Pairing cost (%)	57.3	51.6	40.2	31.1

Table: Performance of the ABE protocol primitives (all the timings are given in 10^3 clock cycles)

Table with the timing costs of the Attribute-based protocol

LSSS ABE Protocol	10^3 clock cycles			
	Attributes in $\mathbb{G}_1$		Attributes in $\mathbb{G}_2$	
	Six attributes	Twenty attributes	Six attributes	Twenty attributes
encryption	$2,384$	$7,150$	$2,921$	$9,129$
key generation	652	$1,699$	$1,326$	$3,994$
decryption ($\Delta = 1$)	$4,606$	$12,776$	$3,515$	$9,528$
overall cost	$7,642$	$21,625$	$7,762$	$22,651$
pairing cost	$4,378$	$11,168$	$3,123$	$7,043$
Pairing cost (%)	57.3	51.6	40.2	31.1

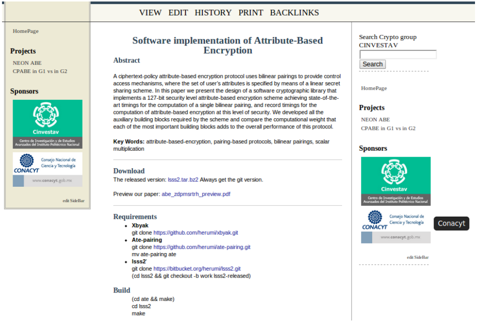
Table: Performance of the ABE protocol primitives (all the timings are given in 10^3 clock cycles)

Table with the timing costs of the Attribute-based protocol

LSSS ABE Protocol	10^3 clock cycles			
	Attributes in $\mathbb{G}_1$		Attributes in $\mathbb{G}_2$	
	Six attributes	Twenty attributes	Six attributes	Twenty attributes
encryption	$2,384$	$7,150$	$2,921$	$9,129$
key generation	652	$1,699$	$1,326$	$3,994$
decryption ($\Delta = 1$)	$4,606$	$12,776$	$3,515$	$9,528$
overall cost	$7,642$	$21,625$	$7,762$	$22,651$
pairing cost	$4,378$	$11,168$	$3,123$	$7,043$
Pairing cost (%)	57.3	51.6	40.2	31.1

Table: Performance of the ABE protocol primitives (all the timings are given in 10^3 clock cycles)

Attribute-based encryption: project web page



Available at: http://sandia.cs.cinvestav.mx/Site/CPABE

Some concrete open problems

1. To produce high performance implementations of more pairing-based protocols

Some concrete open problems

1. To produce high performance implementations of more pairing-based protocols
2. To produce a protected version of the attribute-based protocol presented here

Some concrete open problems

1. To produce high performance implementations of more pairing-based protocols
2. To produce a protected version of the attribute-based protocol presented here
3. To write a compiler tool that allows to provide a high-level description of pairing-based protocols and produces it implementation in Magma and/or C

Some concrete open problems

1. To produce high performance implementations of more pairing-based protocols
2. To produce a protected version of the attribute-based protocol presented here
3. To write a compiler tool that allows to provide a high-level description of pairing-based protocols and produces it implementation in Magma and/or C
4. To produce a high performance symmetric pairing library with fields of large characteristic and elliptic curves with very low embedding degree

Merci-Gracias-Arigato-Thanks
for your attention

borrowed from Quino. Questions?