Pairing-Based Encryption

Implementing pairing-based protocols

Dr. Luis J. Dominguez Perez
ldominguez@computacion.cs.cinvestav.mx

Centro de investigación y de estudios avanzados del IPN Cinvestav

Outline

Outline

Outline

Outline

Motivation

Pairings were originally quite slow compared to other primitives, as a consequence, pairing-based protocols have the fame of being slow.

Nevertheless both the pairing function is competitive, and the pairing-based protocols are faster.

Some pairing-based protocols are impossible with some other primitives, or they are not feasible.

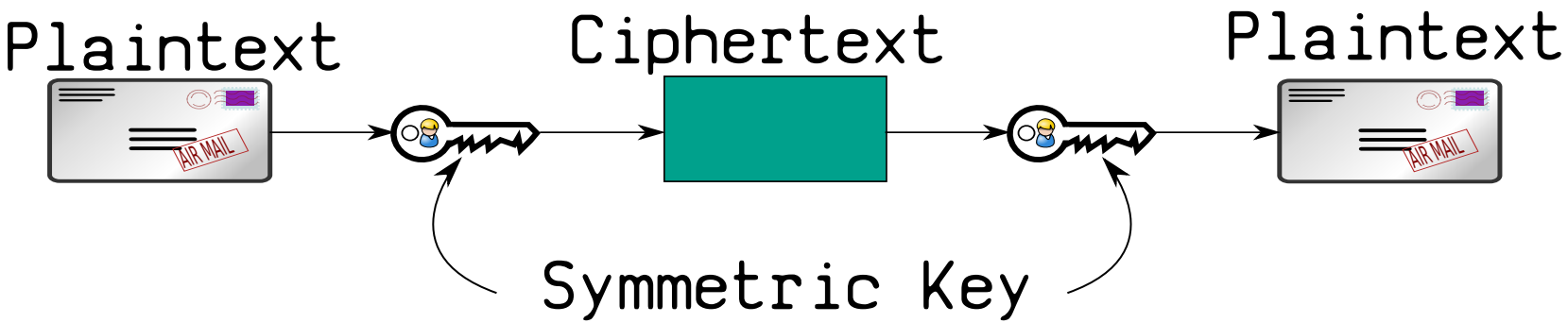
Table of Contents

Cinvestav

Cryptography basics

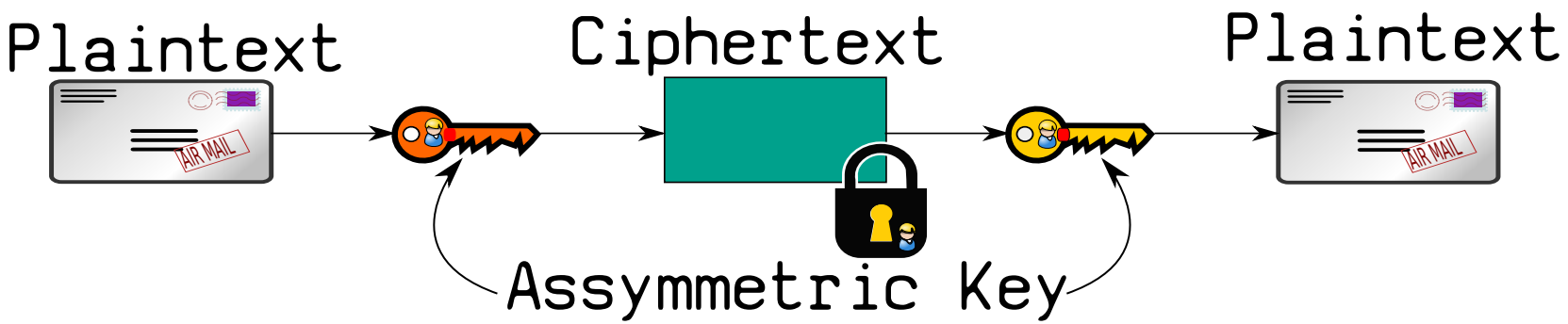
Cryptography supports *symmetric encryption* and *asymmetric encryption* for cryptographic functions:

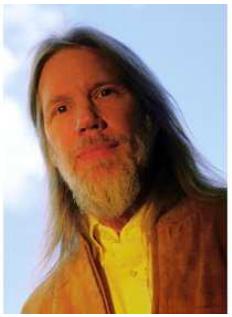
- **Symmetric Encryption.** The same key is used for both encryption and decryption. The key has to be exchanged between the parties in a secure way.



Cryptography basics II

- **Asymmetric Encryption.** Two different, but mathematically related keys are used to encrypt and decrypt the information. Only the public key is needed for other parties. We can broadcast it, there is no need for an exchange protocol.



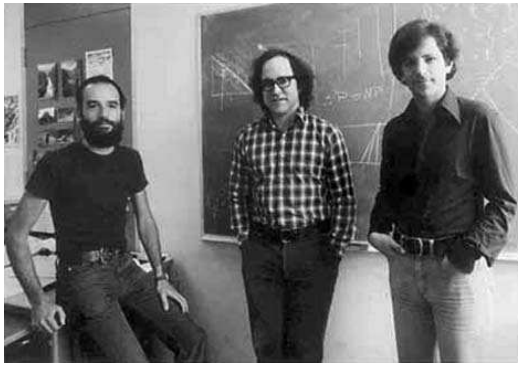



Asymetric Key

The public release of the public key cryptosystem by Diffie and Hellman in 1976 not only created modern cryptography, but also concentrated the Computational Number Theory efforts in this direction.

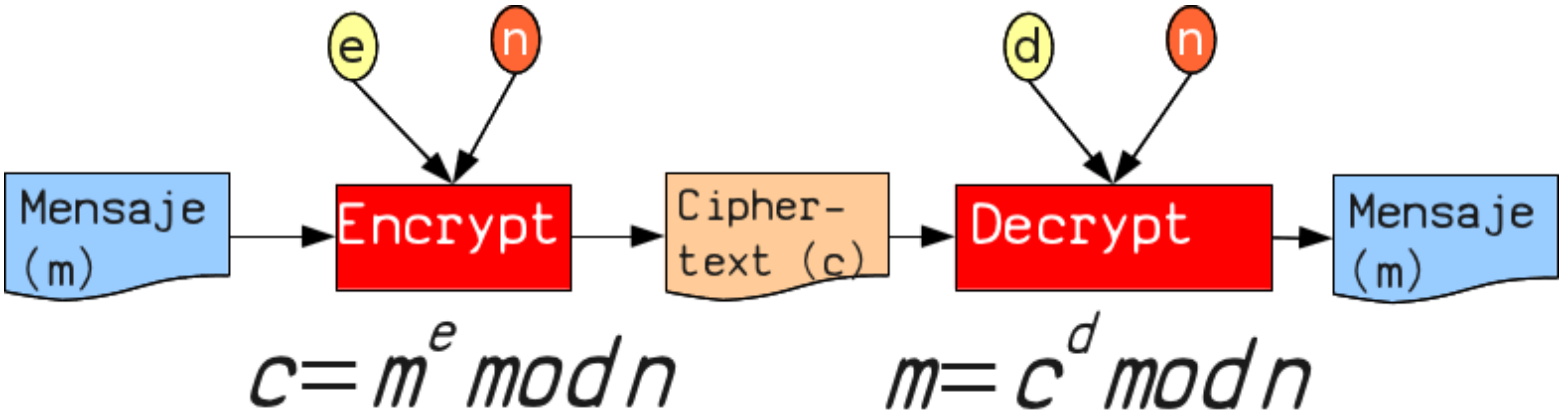
Given a (g, g^x, g^y) what is the value of g^{xy}?

This is meant to be infeasible for sufficiently large values.



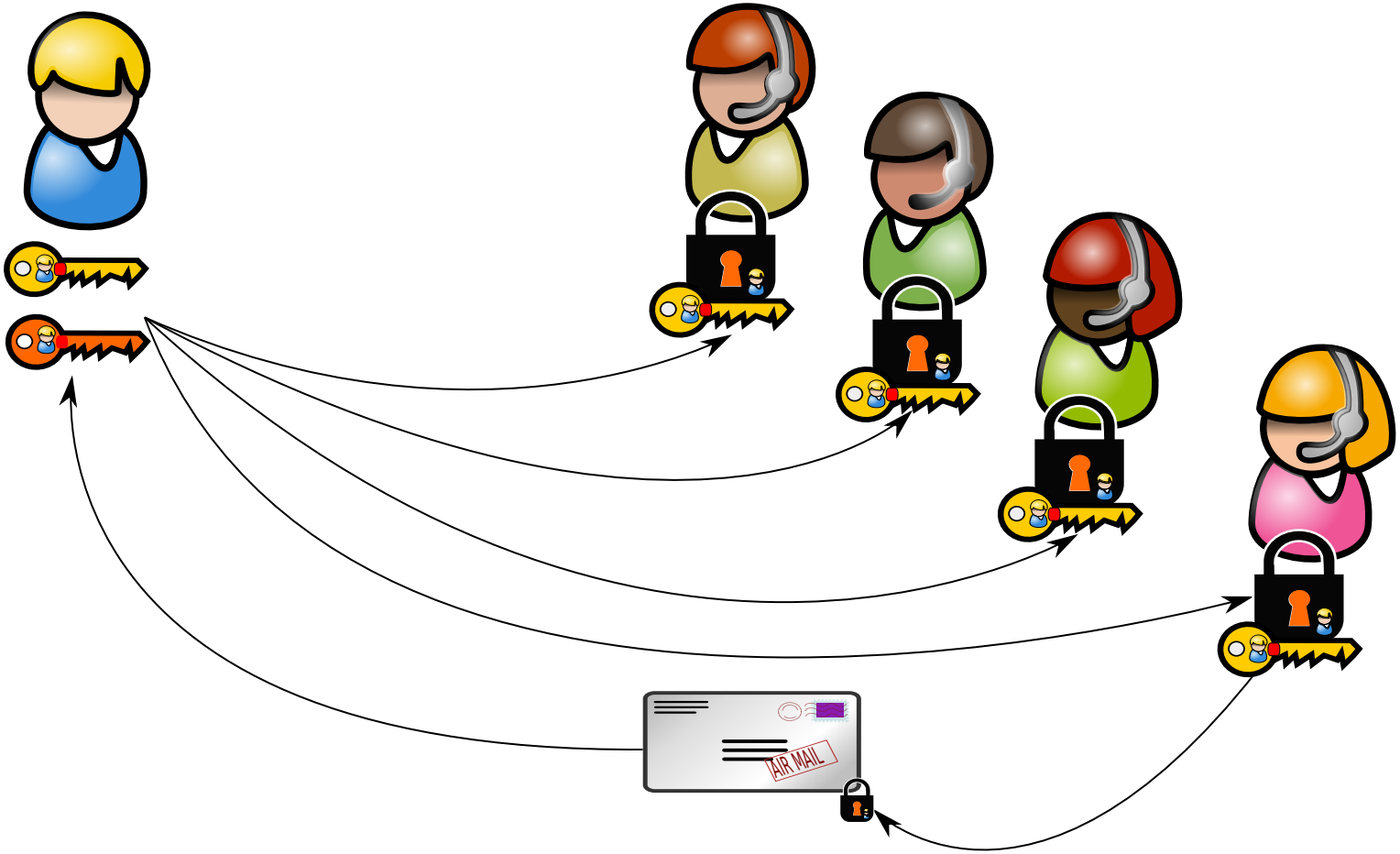
The RSA model

In 1978, the RSA scheme was introduced as the first usable public key cryptosystem. It was based on the problem of factoring large integers.



Cinvestav

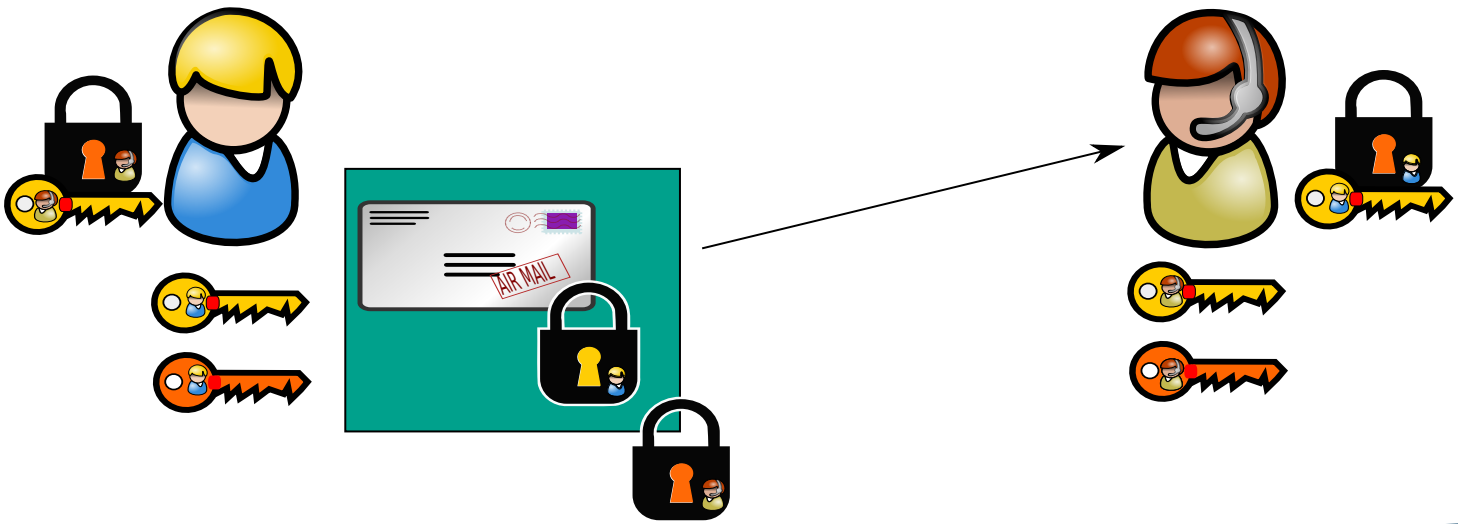
Secret and Public Key



Cinvestav

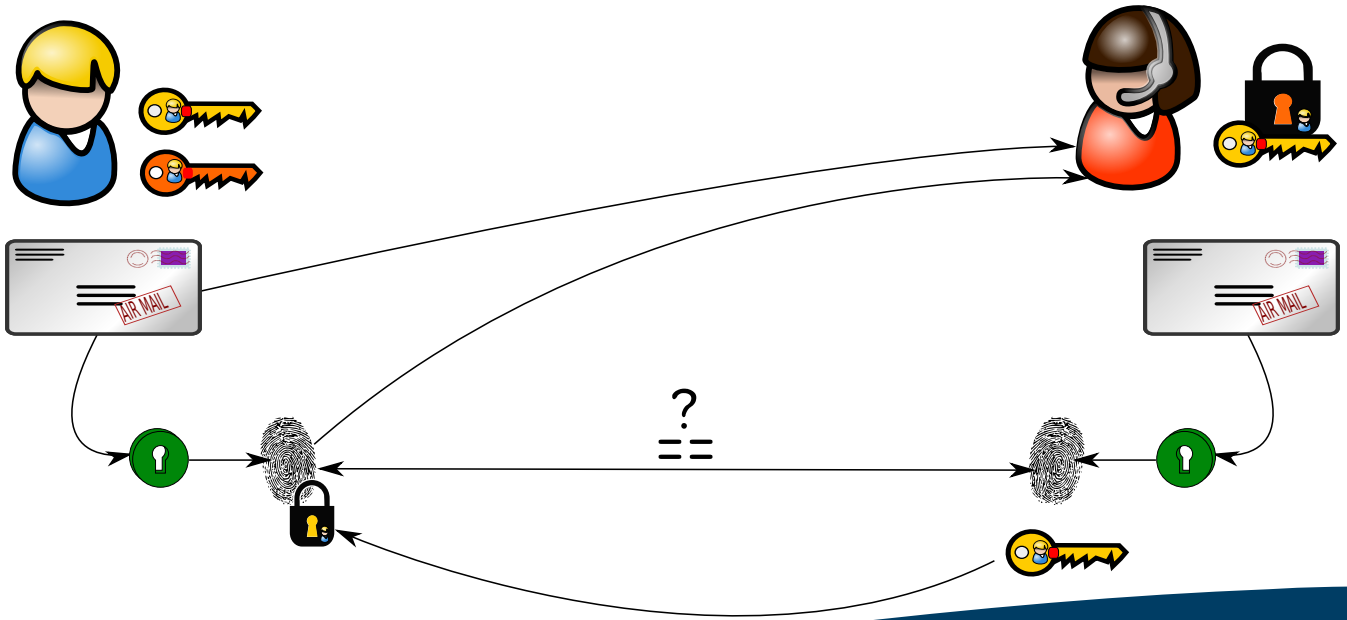
Protecting a message

- Protect the message with our private key so that, everybody will know I sent the message.
- Use recipient's public key so that, only the recipient (owning that private key) reads the message.



Signature of a message

- Sender: Gets the digest of our message
- Sender: Uses our private key on that, send both to the recipient.
- Recipient: Applies the public key of the sender on the message, compare.

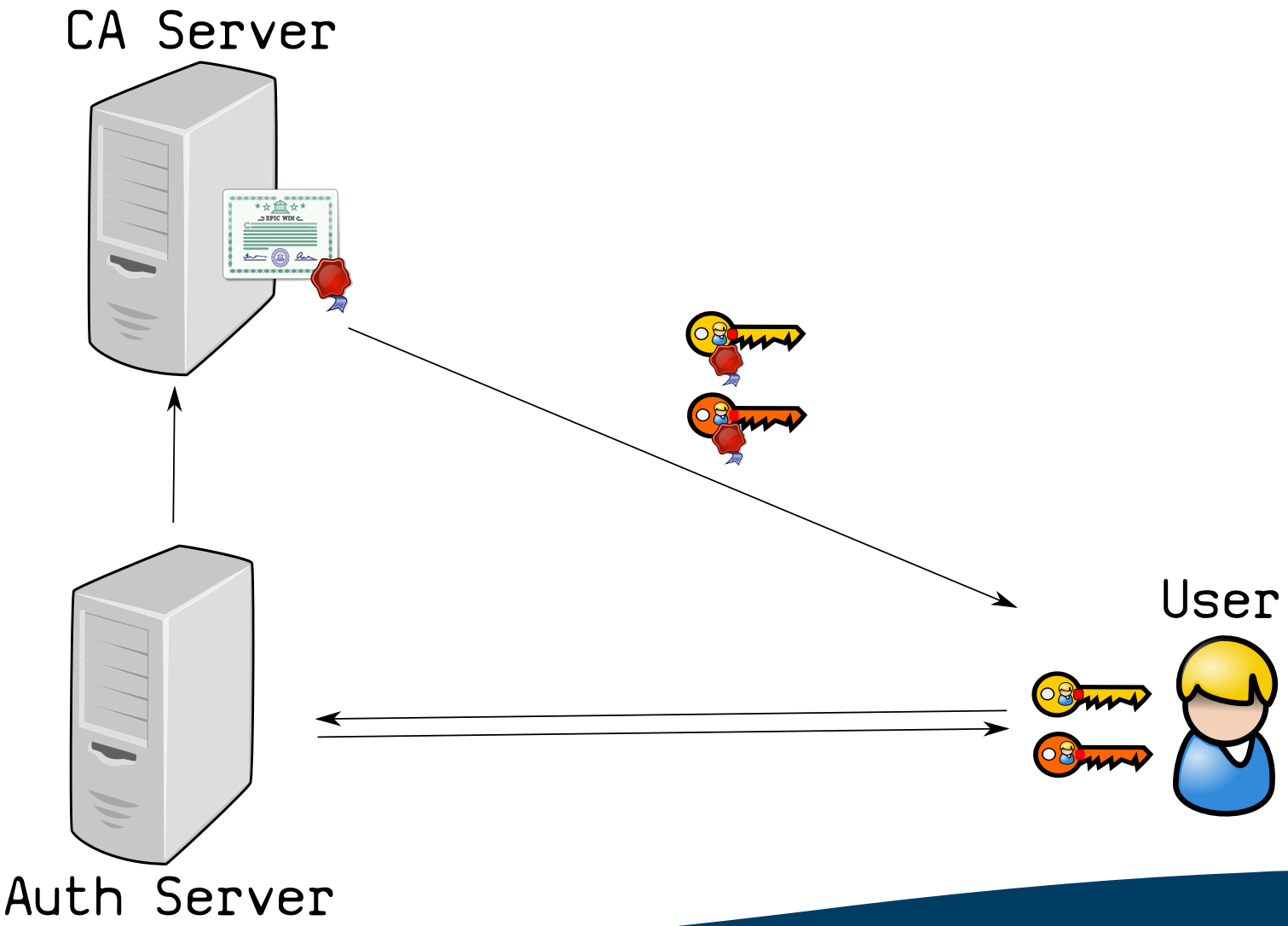


Public Key Infrastructure

- but how do I know the recipient is actually she?
- what about the sender?

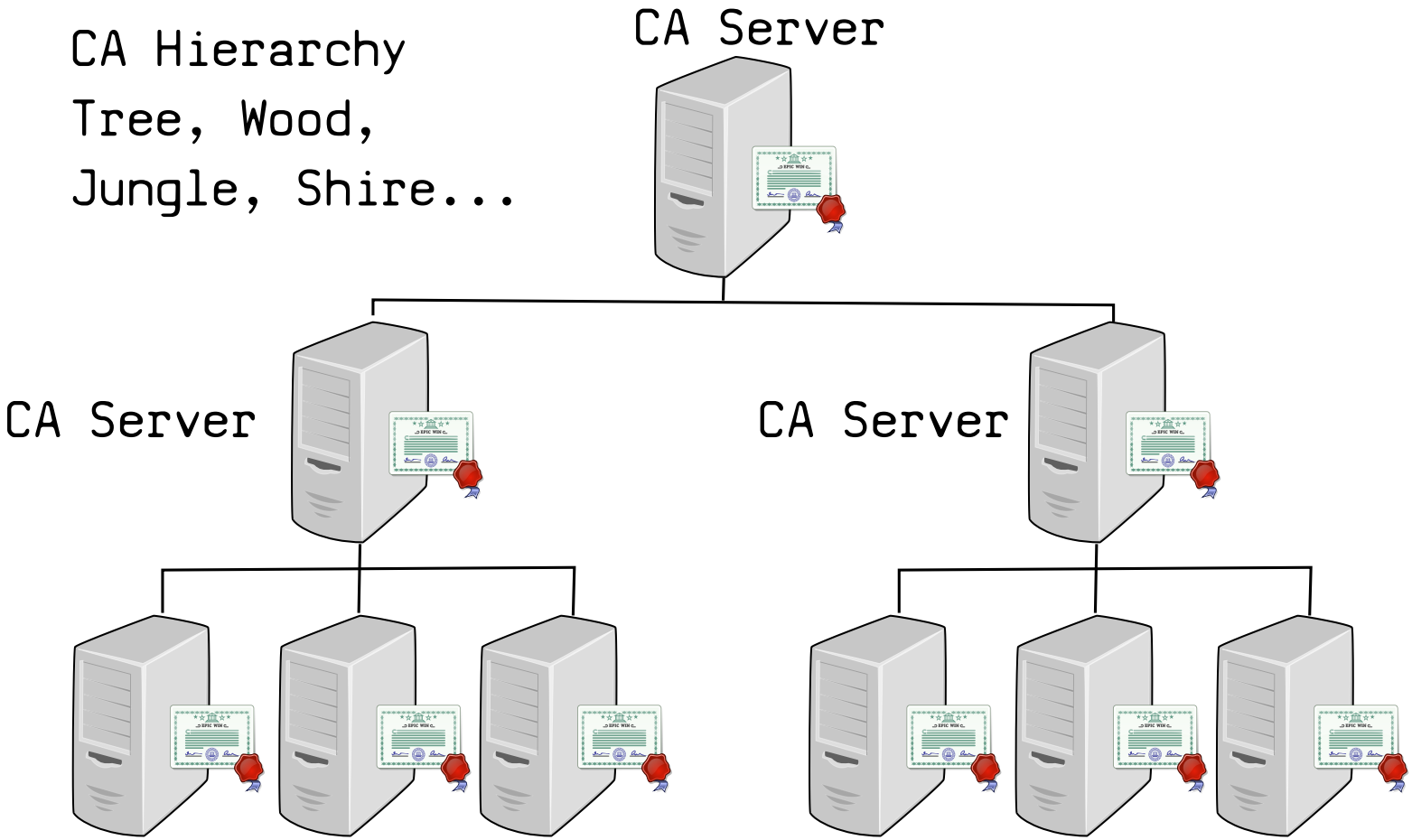
to solve this, we have Public Key Infrastructure...

Public Key Infrastructure



Public Key Infrastructure

CA Hierarchy
Tree, Wood,
Jungle, Shire...



Public Key Infrastructure

CA Hie CA Server

Tree,

Jungle

CA Serve

Even more servers...

Public Key Infrastructure

- To encrypt a message for a user, we get her public key (along with its certificate) in a quite similar process.

Public Key Infrastructure

- To encrypt a message for a user, we get her public key (along with its certificate) in a quite similar process.
- Nice, we have the keys and the certificates stored, how they look like?

They look like a really long sequence of numbers, letters and encoded symbols...

Public Key Infrastructure

- To encrypt a message for a user, we get her public key (along with its certificate) in a quite similar process.
- Nice, we have the keys and the certificates stored, how they look like?

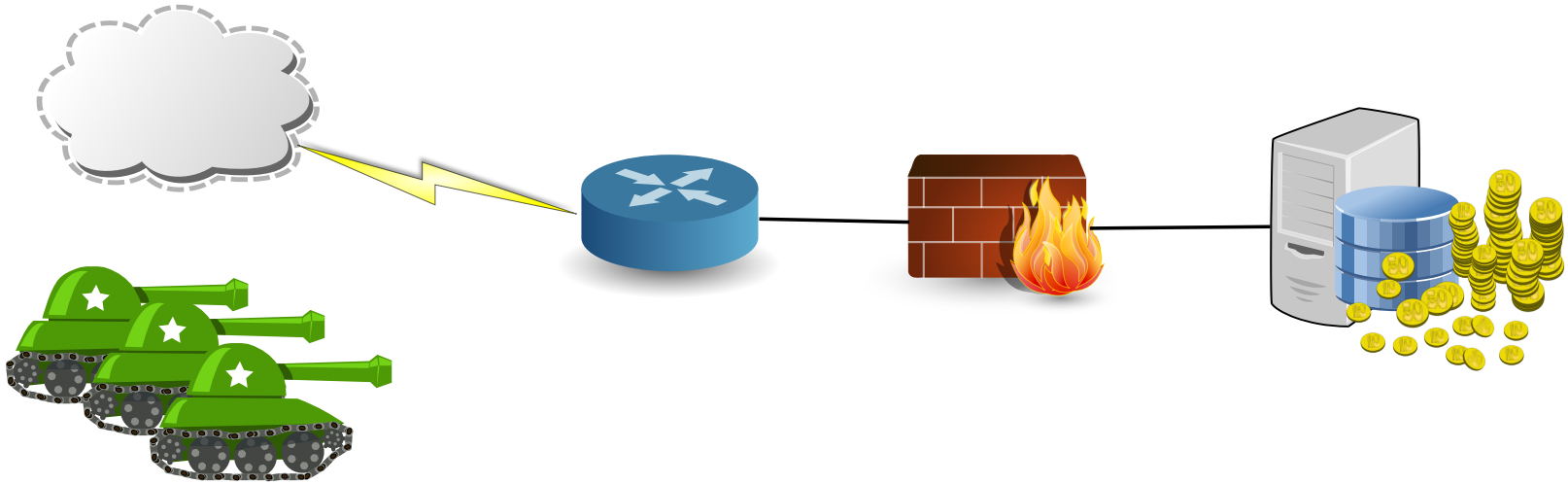
They look like a really long sequence of numbers, letters and encoded symbols... what do you mean with really long?

Cinvestav

Key size and attacks

The main concern about the security of a system, is how long does an attacker need to break it, and how many resources are needed.

The cost of the attack should be balanced with respect to the value of the treasure:



Key size and attacks II

There are many attacks against the cryptosystems, and depending on how long they will take to successfully break a system with a given key size is the security associated to them.

For a symmetric key, the minimum recommended security level by the NIST is to have a key of around 128 bits, however, for an RSA asymmetric key to take as long as a symmetric key of 128 bits to be broken is of around 3072 bits!!!

Key size and attacks III

NIST states that an 80-bit symmetric key is equivalent to a 160-bit one using discrete logs subgroups and elliptic curve groups. This is defined as a 80-bit security level, and it is not recommended for use after 2012. An 128-bit security level is recommended therefore after that year.

Equivalent symmetric key size		80	112	128	192	256
NIST	RSA	1024	2048	3072	7680	15360
	EC	160	224	256	384	512
ECRYPT	RSA	1248	2432	3248	7936	15424
	EC	160	224	256	384	512

More complains

- and what happens if the user is not explicitly in the system?

More complains

- and what happens if the user is not explicitly in the system?
- or what about giving access to a group of users with the same characteristics?

Cinvestav

More complains

- and what happens if the user is not explicitly in the system?
- or what about giving access to a group of users with the same characteristics?

Wait!

More complains

- and what happens if the user is not explicitly in the system?
- or what about giving access to a group of users with the same characteristics?

Wait! Why would someone would like to encrypt something for a non-existant user?...

More complains

- and what happens if the user is not explicitly in the system?
- or what about giving access to a group of users with the same characteristics?

Wait! Why would someone would like to encrypt something for a non-existant user?... That's non-sense!

More complains

- and what happens if the user is not explicitly in the system?
- or what about giving access to a group of users with the same characteristics?

Wait! Why would someone would like to encrypt something for a non-existant user?... That's non-sense!

or is it?

What's next?

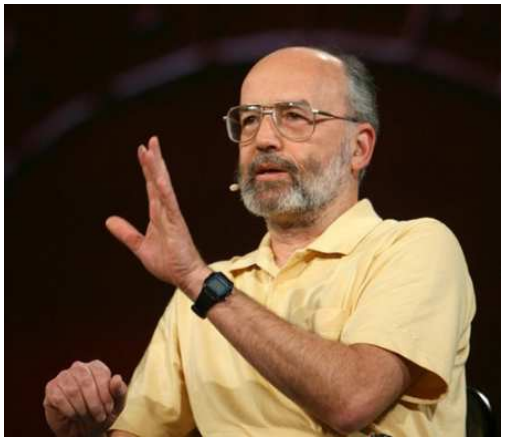
Are there anything we can do to solve all of this?

What's next?

Are there anything we can do to solve all of this?

Yes! We can use the Pairing-Based Cryptography.

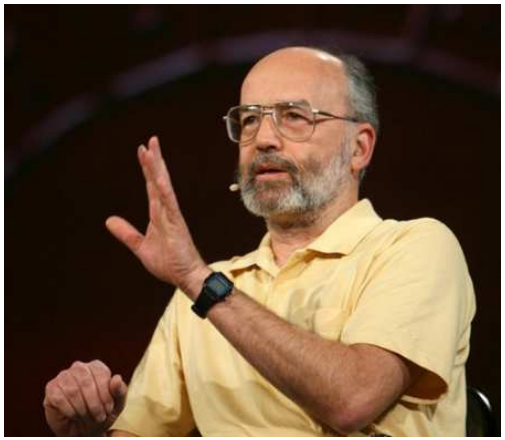
Table of Contents



How does it started?

In 1984, Shamir posed a challenge:

"create a cryptographic system that permits any two users to comunicate securely and to verify each other's signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party."



How does it started?

In 1984, Shamir posed a challenge:

"create a cryptographic system that permits any two users to comunicate securely and to verify each other's signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party."

This sounds impossible!

How does it started? 2

However, in 2001, Boneh and Franklin, solved this challenge using cryptographic pairings. They presented what it is now called Identity-Based Encryption.

How does it started? 2

However, in 2001, Boneh and Franklin, solved this challenge using cryptographic pairings. They presented what it is now called Identity-Based Encryption.

... also, in 2000, Antoine Joux presented a breaking-through paper involving pairings, but we are focusing in this talk on Identity-Based Encryption.

Pairing-Based Cryptography

Identity-Based Encryption is a type of the Pairing-Based Encryption, this is, we use some cryptographic function called the pairing.

In essence, a cryptographic pairing is a particular function of groups over elliptic curves.

$$\langle \cdot, \cdot \rangle : M \times M \longrightarrow R$$

The bilinear pairing can be used as a primitive to build cryptosystems with certain functionality. Examples of use:

The bilinear pairing can be used as a primitive to build cryptosystems with certain functionality. Examples of use:

- Short signatures schemes,

The bilinear pairing can be used as a primitive to build cryptosystems with certain functionality. Examples of use:

- Short signatures schemes,
- Identity-Based Encryption,

The bilinear pairing can be used as a primitive to build cryptosystems with certain functionality. Examples of use:

- Short signatures schemes,
- Identity-Based Encryption,
- Attribute-Based Encryption,

The bilinear pairing can be used as a primitive to build cryptosystems with certain functionality. Examples of use:

- Short signatures schemes,
- Identity-Based Encryption,
- Attribute-Based Encryption,
- and other protocols already deployed.

Some protocols are impossible with currently deployed technology, in other cases, they are faster.

Example of PBC

Identity-Based Encryption case:

- Enables any pair of users to communicate securely and to verify each others' signatures **without exchanging** private or public keys;
- Needs **no key server repositories**;
- Requires a trusted server for key generation **only**.
- **No certificate required** to bind the public key to the identity.

Implementation issues...

Pairing-Based Cryptography has become relevant in industry.

Although there are plenty of applications, however efficiently implementing the pairings function is often difficult as it requires more knowledge than previous cryptographic primitives.

There are many implementation issues just with the primitive itself!

Cinvestav

... implementation issues

- **Non-familiar** technology;
- Lack of **programming framework**;
- More **difficult to understand** compared to the already deployed technology;
- **Unavailability** of implementations with novel (faster) computing methods;
- Complex area.

Depending on the scenario, a developer must choose from a selection of parameters and apply the corresponding optimizations for efficiency...

What to do when... ?

- **bandwidth** use is expensive;
- **low memory** is available;
- a **slow** processor is used (old);
- a **small** processor (in bits) is the only option;
- we have a **Desktop** environment;
- we have a device with **multiprocessors**;
- a **higher security** is required;

Some basic operations that are cheap in some environments are expensive in others!

Protocol primitives

The operations involved in a Pairing-Based protocol are:

- The pairing function
- Elliptic Curve point addition and point doubling
- Scalar-point multiplication
- exponentiation
- hash onto a curve
- hash into a subgroup
- matrix conversion
- boolean function analysis...

Many more!

Some background

Let do a bit of maths...

Scalar-point multiplication

Let P be a point in a curve E and $n \in \mathbb{Z}$, $n \geq 0$. Define $[n]P = P + P + \cdots + P$. The order of the point P is the smallest n such that $[n]P = \mathcal{O}$.

Denote $< P >$ the group generated by P. In other words,

$$< P >= \{\mathcal{O}, P, P+P, P+P+P, \ldots\}$$

Let $Q \in < P >$. **Given Q, find n such that $Q = [n]P$ is hard.**

Applying the algorithm

The traditional method for computing the scalar-point multiplication is the Double-and-Add method.

Algorithm 1 Traditional scalar-point multiplication

Input: Positive integer k in base 2 representation, a point P.
Ouput: $[k]P$

1: $Q \leftarrow 0$
2: **for** $i = l - 1$ downto 0 **do**
3: $\quad Q \leftarrow [2]Q$
4: $\quad$ **if** $k_i = 1$ **then**
5: $\quad\quad Q \leftarrow Q + P$
6: $\quad$ **end if**
7: **end for**
8: **return** Q

Speeding up

Generic method to speed up the exponentiation in this context:

- Precomputation
- Addition chains whenever the scalar is known
- Windowing techniques
- Simultaneous multiple exponentiation techniques.

Replacing the binary representation of the scalar into one with fewer non-zero terms.

Cinvestav

Speeding up II

Curve specific methods:

- A field defined with a (pseudo-)Mersenne prime.
- Field construction using small irreducible polynomials
- Point representation with fast arithmetic
- EC with special properties.

Pairing definition

A **pairing** is a map: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

These groups are finite and cyclic. $\mathbb{G}_1$ and $\mathbb{G}_2$ are additively-written and at least one is of prime order r. $\mathbb{G}_T$, is multiplicatively-written and of order r.

Properties:

- Bilinearity
- Non-degeneracy
- Efficiently computable

Cinvestav

Pairing properties

Properties:

- Bilinearity

$$e(P + P', Q) = e(P, Q) \times e(P', Q)$$

$$e(P, Q + Q') = e(P, Q) \times e(P, Q')$$

Cinvestav

Pairing properties

Properties:

- Bilinearity
  $e(P + P', Q) = e(P, Q) \times e(P', Q)$
  $e(P, Q + Q') = e(P, Q) \times e(P, Q')$
- Non-degeneracy
  $\forall P \in \mathbb{G}_1,\ P \neq \mathcal{O}$: $\exists Q \in \mathbb{G}_2$ s.t. $e(P, Q) \neq 1$
  $\forall Q \in \mathbb{G}_2,\ Q \neq \mathcal{O}$: $\exists P \in \mathbb{G}_1$ s.t. $e(P, Q) \neq 1$ $\quad$ $e(P, Q) \neq 1$

Cinvestav

Pairing properties

Properties:

- Bilinearity

  $e(P + P', Q) = e(P, Q) \times e(P', Q)$

  $e(P, Q + Q') = e(P, Q) \times e(P, Q')$

- Non-degeneracy

  $\forall P \in \mathbb{G}_1,\ P \neq \mathcal{O}$: $\exists Q \in \mathbb{G}_2$ s.t. $e(P, Q) \neq 1$

  $\forall Q \in \mathbb{G}_2,\ Q \neq \mathcal{O}$: $\exists P \in \mathbb{G}_1$ s.t. $e(P, Q) \neq 1$ $\qquad$ $e(P, Q) \neq 1$

- Efficiently computable

(Ab)Using the pairing

The most important property of a pairing is:

$$e([a]Q, [b]P) = e([b]Q, [a]P) = e(Q, [ab]P) = e(Q, P)^{ab}$$

where $Q \in \mathbb{G}_2$, $P \in \mathbb{G}_1$, and the result is in $\mathbb{G}_T$.

In our context, the $\mathbb{G}_2$ group is larger than $\mathbb{G}_1$. The group $\mathbb{G}_T$ is also larger and has a different set of operations.

Cinvestav

(Ab)Using the pairing II

- Since $\mathbb{G}_2$ is larger than $\mathbb{G}_1$, it is wise to exchange operations from one group to the other.
- $\mathbb{G}_T$ is significantly larger and has a different set of operations, we also try to avoid it, but we keep it handy, because...

Cinvestav

(Ab)Using the pairing II

- Since $\mathbb{G}_2$ is larger than $\mathbb{G}_1$, it is wise to exchange operations from one group to the other.
- $\mathbb{G}_T$ is significantly larger and has a different set of operations, we also try to avoid it, but we keep it handy, because...
- An operation in $\mathbb{G}_T$ is cheaper than computing the pairing itself.

In short, we use the groups at will.

Using the pairing III

In each pairing-based protocol, we have to use the pairing function in a different way.

In some cases, we have to compute several-to-many pairings, in other cases, the pairings have the same parameters, in other cases, we have to add the results.

For this, we design the pairing function to be a:

- Multipairing
- Known-point pairing
- or we mix-up several parameters into a single pairing.

Scalar-point multiplication in $\mathbb{G}_1$

Gallant, Lambert and Vanstone find out a method to do the scalar-point multiplication by breaking the scalar n into two smaller scalars, under special situations, the operation is faster.

Let E be a curve defined over the field $\mathbb{F}_p$ with zero denoted by $\mathcal{O}$.

An endomorphism of E is a rational map $\phi : E \rightarrow E$ satisfying $\phi(\mathcal{O}) = \mathcal{O}$.

There exists an endomorphism such that the following holds:

$$[n]P = [n_0]P + [n_1]\lambda P$$

where, $|n_i| \approx |\sqrt{n}|$.

Scalar-point multiplication in $\mathbb{G}_2$

Galbraith and Scott, showed a technique for generalizing the GLV method for higher powers of the endomorphism for the groups $\mathbb{G}_2$ and $\mathbb{G}_T$.

To get an m-dimensional expansion

$$n \equiv n_0 + n_1\lambda + \cdots + n_{m-1}\lambda^{m-1} \pmod{r}$$

of $[n]P$, one must decompose n with powers of λ sufficiently different and modulo r, an associated prime number to the curve.

Exponentiation in $\mathbb{G}_T$

In the case of exponentiation in $\mathbb{G}_T$, an efficiently computable endomorphism is also available.

We decompose the operation as:

$$e^n = e^{n_0} \cdot e^{n_1^p} \cdot e^{n_2^{p^2}} \cdots e^{n_1^{p^{m-1}}}$$

where $e \in \mathbb{G}_T$, $n \in \mathbb{Z}_r$, m is the degree of the decomposition, and the exponentiation to the p is done using the Frobenius endomorphism method.

Cinvestav

LSSS matrix

A secret scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if

- The shares of each party form a vector over $\mathbb{Z}_p$
- There exists a matrix A called the share generating matrix for Π. The matrix A is an $m \times n$ matrix. For all $i = 1, \ldots, m$, the i^{th} row of A is labeled by a party $\rho(i)$ (ρ is a function from $\{i, \ldots, \}$ to $\mathcal{P}$.

LSSS example

The LSSS transformation algorithm takes the boolean formula F, a user defined policy, into a LSSS matrix. For example if the user wants to encypher data for: "a Mexican person, from the Cinvestav, and with either a Doctoral or Master's degree", then the corresponding formula would be:

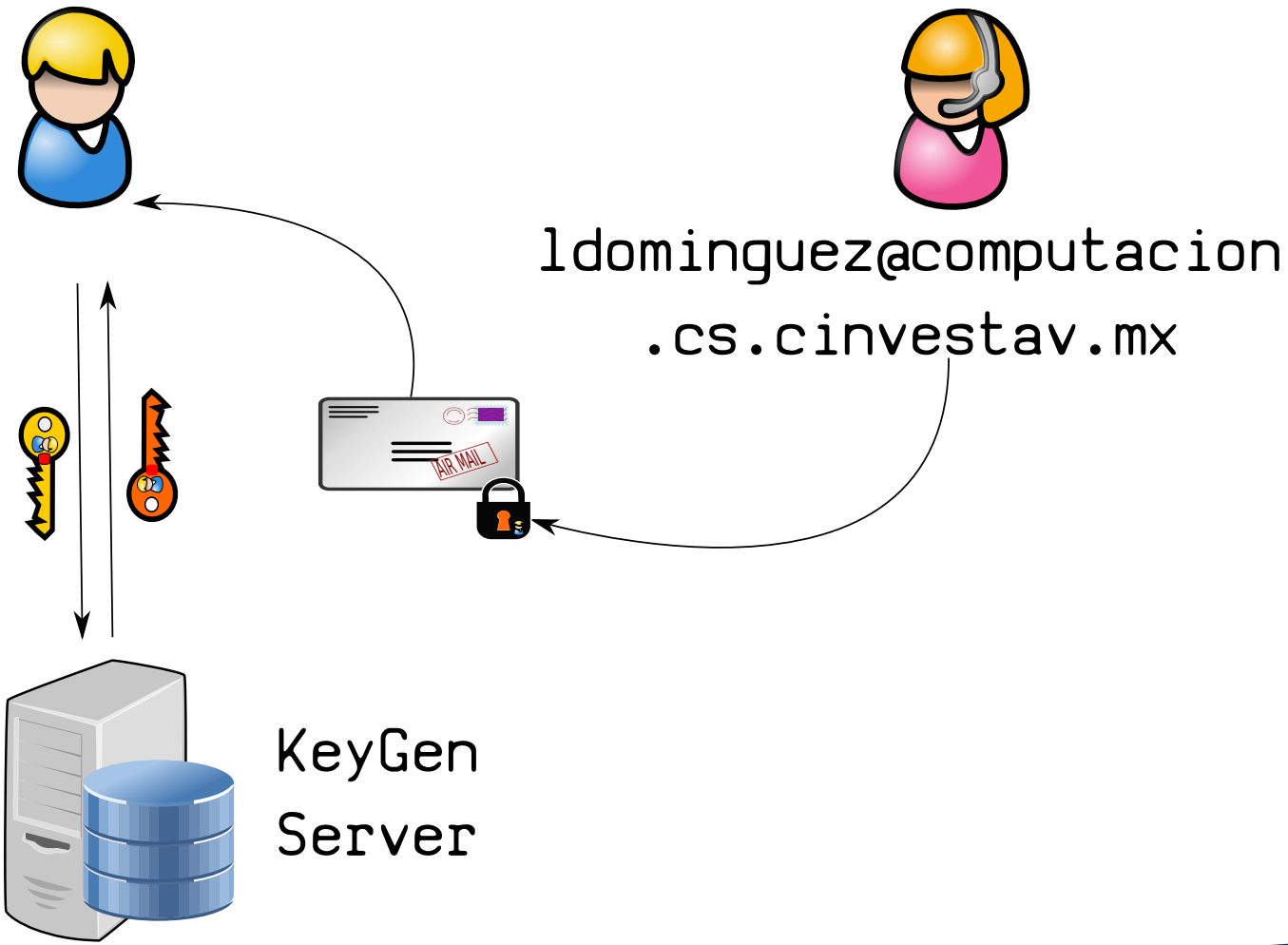
$$\mathrm{F} = \text{“(Mexican,Cinvestav,(Doctor,Master,1),3)”}$$

The transformation would be something like:

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 3 & 9 \end{pmatrix} \begin{matrix} \textit{Mexicano} \\ \textit{Cinvestav} \\ \textit{Doctor} \\ \textit{Master} \end{matrix}$$

Table of Contents

Encryption for an identity



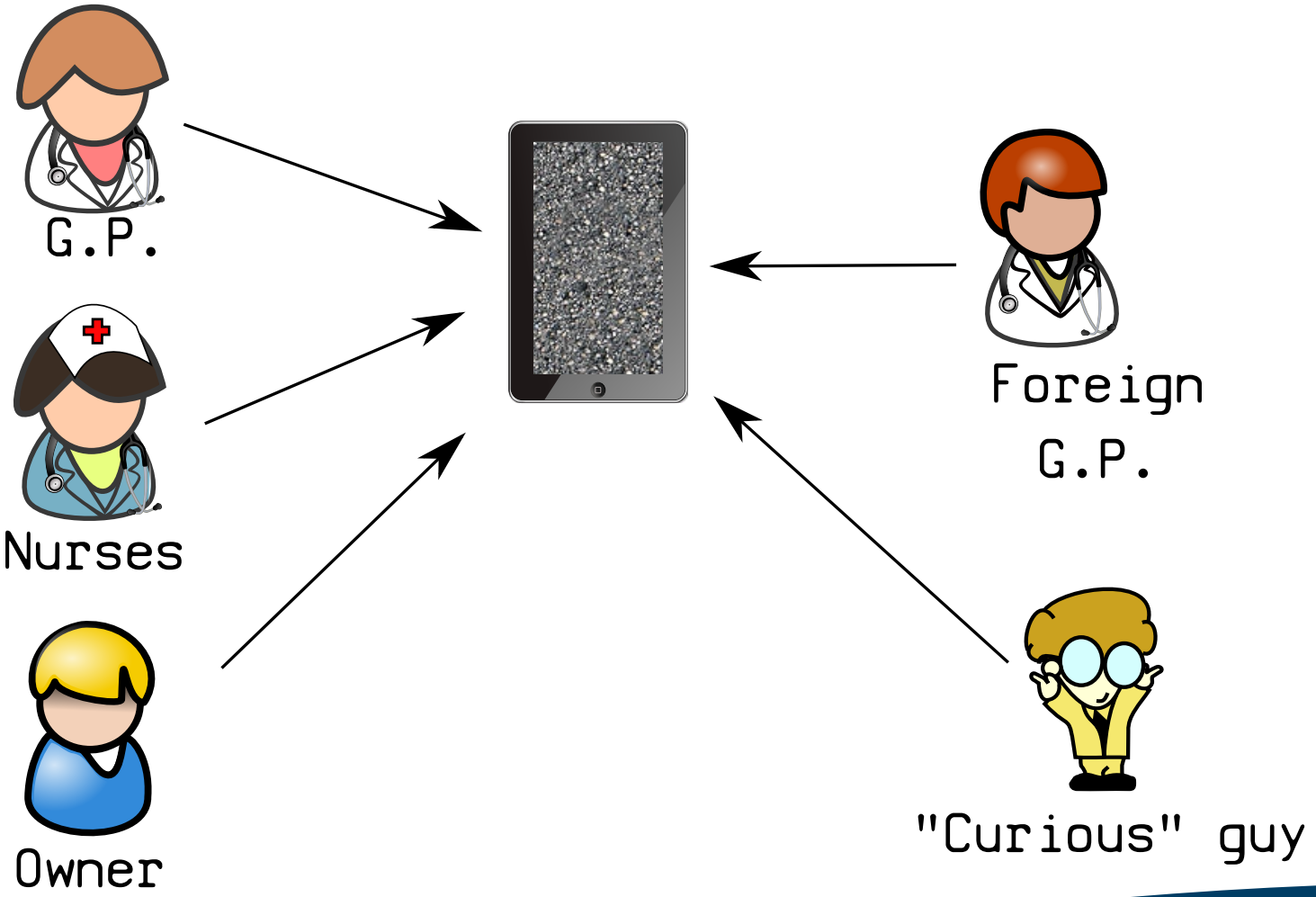
Attribute-Based Encryption

Fuzzy Identity-Based Encryption. Also known as Attribute-based encryption.

- An identity is a set of attributes
- An entity is valid if it presents a minimum number of attributes
- Better for sharing a small secret: a symmetric key.

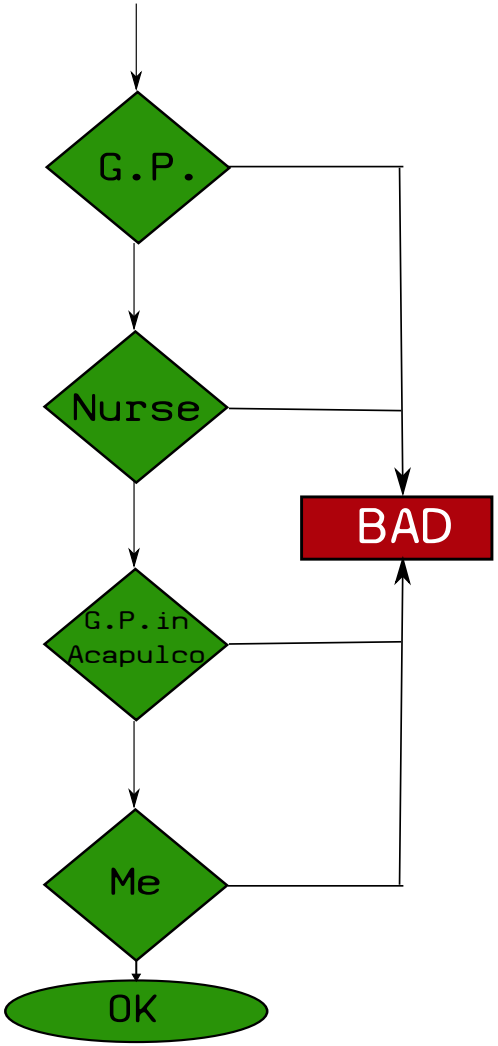
Cinvestav

Attribute examples



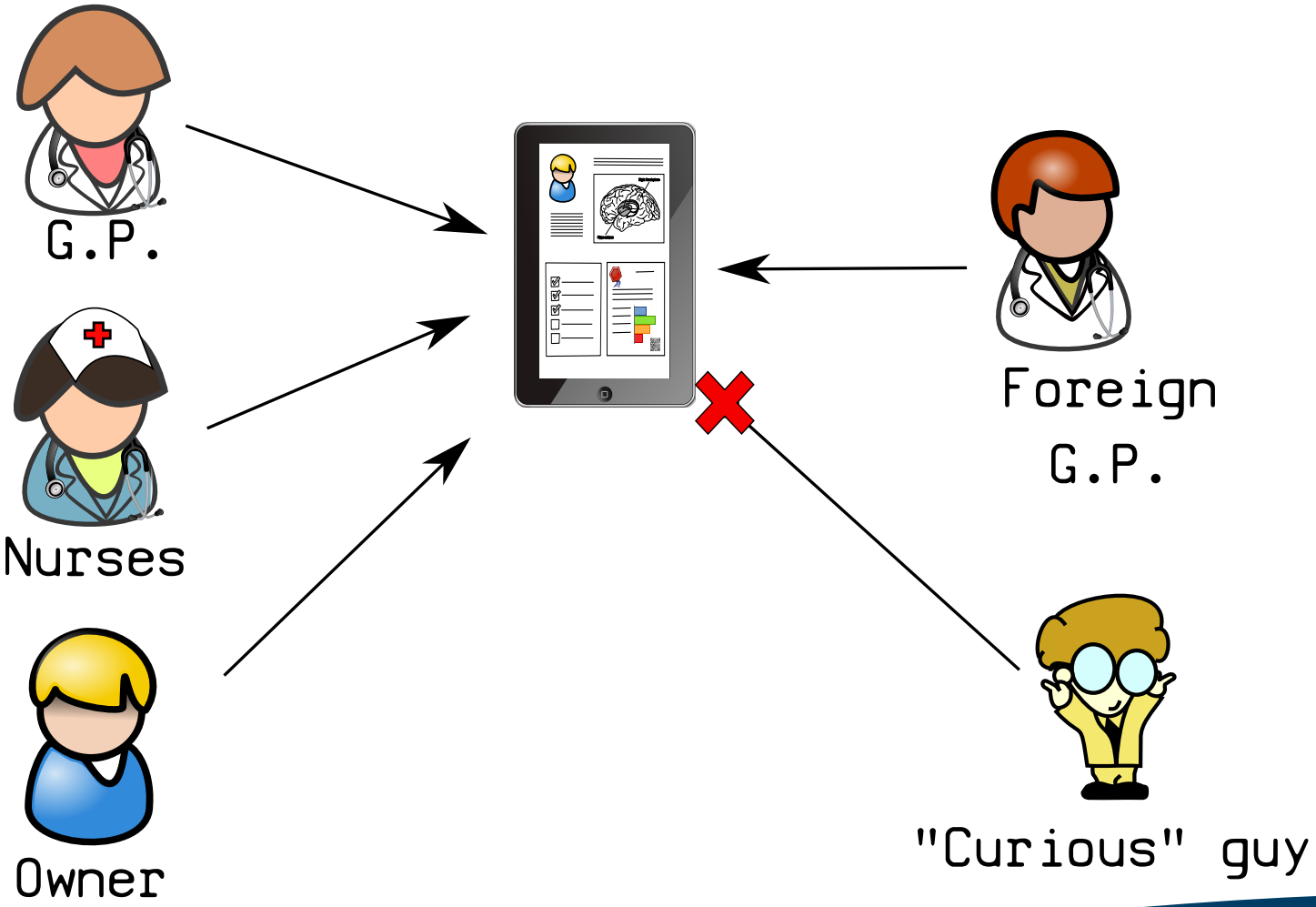
Cinvestav

Attribute examples



Cinvestav

Attribute examples



Cinvestav

Water's CPABE

Setup. In the CPABE protocol, we have the typical four cryptographic protocol steps (setup, key generation, encryption, and decryption), plus a fifth optional step: delegation.

Public Key

$\mathsf{PK} = ([\alpha]P, [d]P, Q, [d]Q, \hat{e}^{\alpha}), \{H_1 \dots H_i\})$ $\quad\quad$ $\mathsf{MK} = ([\alpha]P)$

where $P \in \mathbb{G}_1$; $Q \in \mathbb{G}_2$;$\hat{e} = e(Q, P)$; $d, \alpha \in_R \mathbb{Z}_r$; and H_i are the known attributes represented in $\mathbb{G}_1$ and $\mathbb{G}_2$.

Water's CPABE 1/3

Encryption. The user defines his access policy by providing a boolean formula F, then this string is transformed into a $m \times n$ LSSS matrix $\mathcal{S}$, we encrypt the *Message* K. The CypherText contains:

Ciphertext

CT $= \mathcal{S}, C, C_d$, (for i=1 ... m): C_i, D_i

where $d \in_R \mathbb{Z}_r$, $C = K\hat{e}^d$, $C_d = [d]Q$, and $C_i = [\lambda_i]P_\alpha - [s_i]H_{\rho(i)}$, $D_i = [s_i]Q$). In practice, the K is an AES key or another secret.

Cinvestav

Water's CPABE 2/3

Key Generation. Take the MK, and the set of attributes $\mathcal{S}$ we would like the key to generate.

Secret Key

$SK = (K, L, K_i)$

with $K = P_\alpha + [t]P_a$, $L = [t]Q$, and $\forall i \in \mathcal{S} : K_i = [t]H_i$.

Delegation. Here, we only take a subset of elements in $\mathcal{S}$ and generate the key for that specific subset.

Water's CPABE 3/3

Decrypt.

- $\Delta \leftarrow \frac{Det(S')}{GCD(Det(S'),\bar{\omega})}$
- For $i \in [1..m]$ $C_i \leftarrow [\omega_i]C_i$, $K_i \leftarrow [\omega_i]K_i$
- $K_{AES} = C \cdot \left(e(C_d, -[\Delta]K) \cdot e(L, \sum_{i \in A'} C_i) \cdot \prod_{i \in A'} e(D_i, K_i) \right)^{\Delta^{-1}}$

where $A' \leftarrow \mathcal{S} \cup SK$, ω_i is a random number resulting from transforming the policy into the $\mathcal{S}$ matrix.

Operations in Water's CPABE

Step	Operation count
Encrypt:	$1\ \mathbb{G}_T$ expo K $+ (n+1)\ \mathbb{G}_2$ S.M. K $+ (2n)\ \mathbb{G}_1$ S.M. K
Keygen:	$(n+1)\ \mathbb{G}_1$ S.M. K $+\ \mathbb{G}_2$ S.M. K
Decrypt:	$(\Delta = 1)$: 1 MultiPairing $+ (2n)\ \mathbb{G}_1$ S.S.M. U $(\Delta \neq 1)$: 1 MultiPairing $+ (2n)\ \mathbb{G}_1$ S.S.M. U $+\ \mathbb{G}_T$ expo U

Table: Operation count of the steps in the protocol

Boneh's short signatures

Boneh's short signatures are based on the mathematical problem:

Given $(P, [a]P, Q, [b]Q)$, it is hard to decide if $a = b$

The computational variant of this hard problem is:

Given $(P, Q, [n]Q)$, compute $[n]P$

Boneh, Lynn and Shacham constructed a short signature scheme based on this problem as follows:

Cinvestav

... the steps

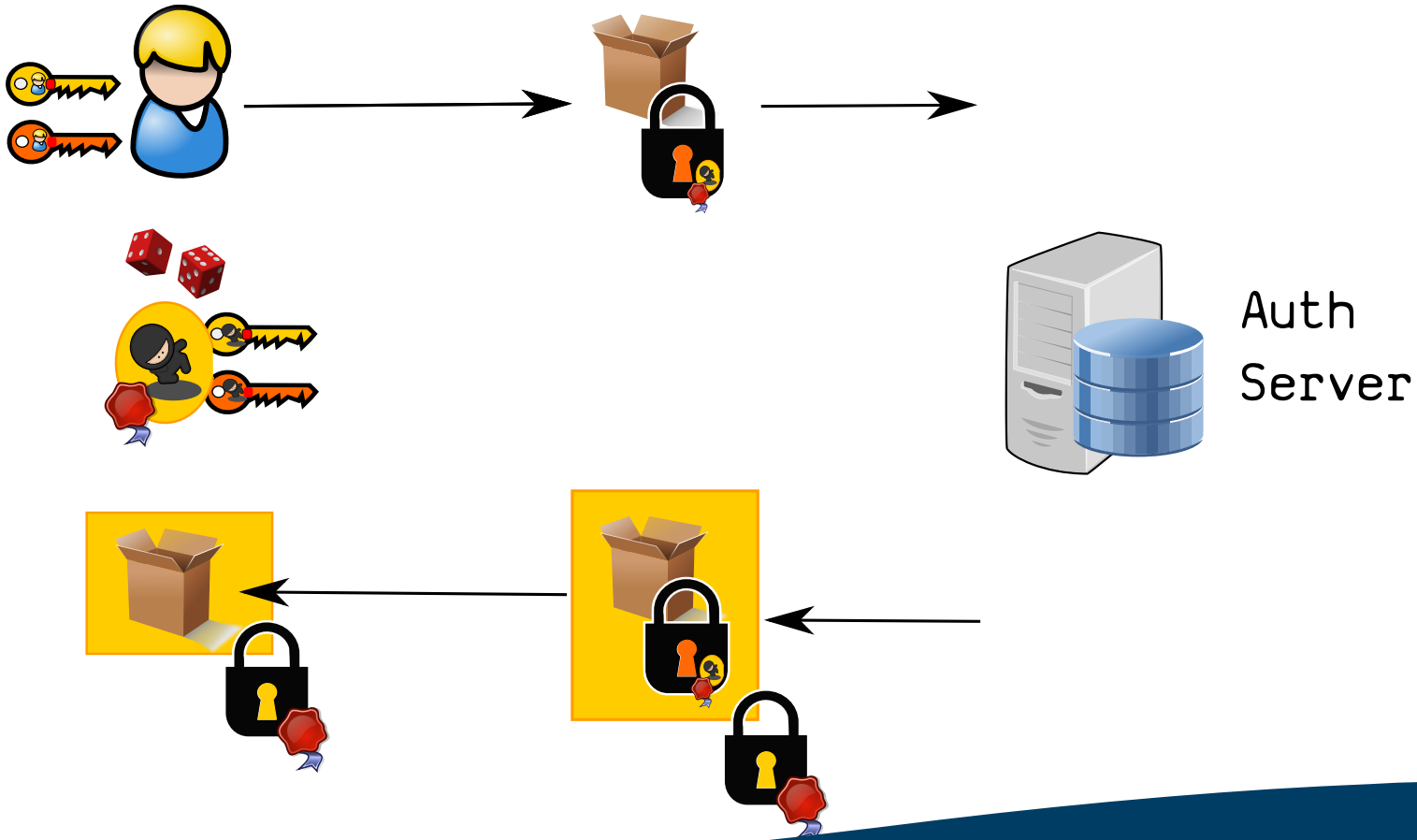
Key generation. Choose $n \in_R \mathbb{Z}_r$, set $R \leftarrow [n]Q$. The public key is: Q, R. The secret key is n

Sign. Map to a point the message to sign as P_M, set $S_M \leftarrow [n]P_M$. The signature is the x-coordinate of S_M.

Verify. Given the x-coordinate of S_M, find $\pm S$. Decide: $e(Q, S) \stackrel{?}{=} e(R, h(M))$

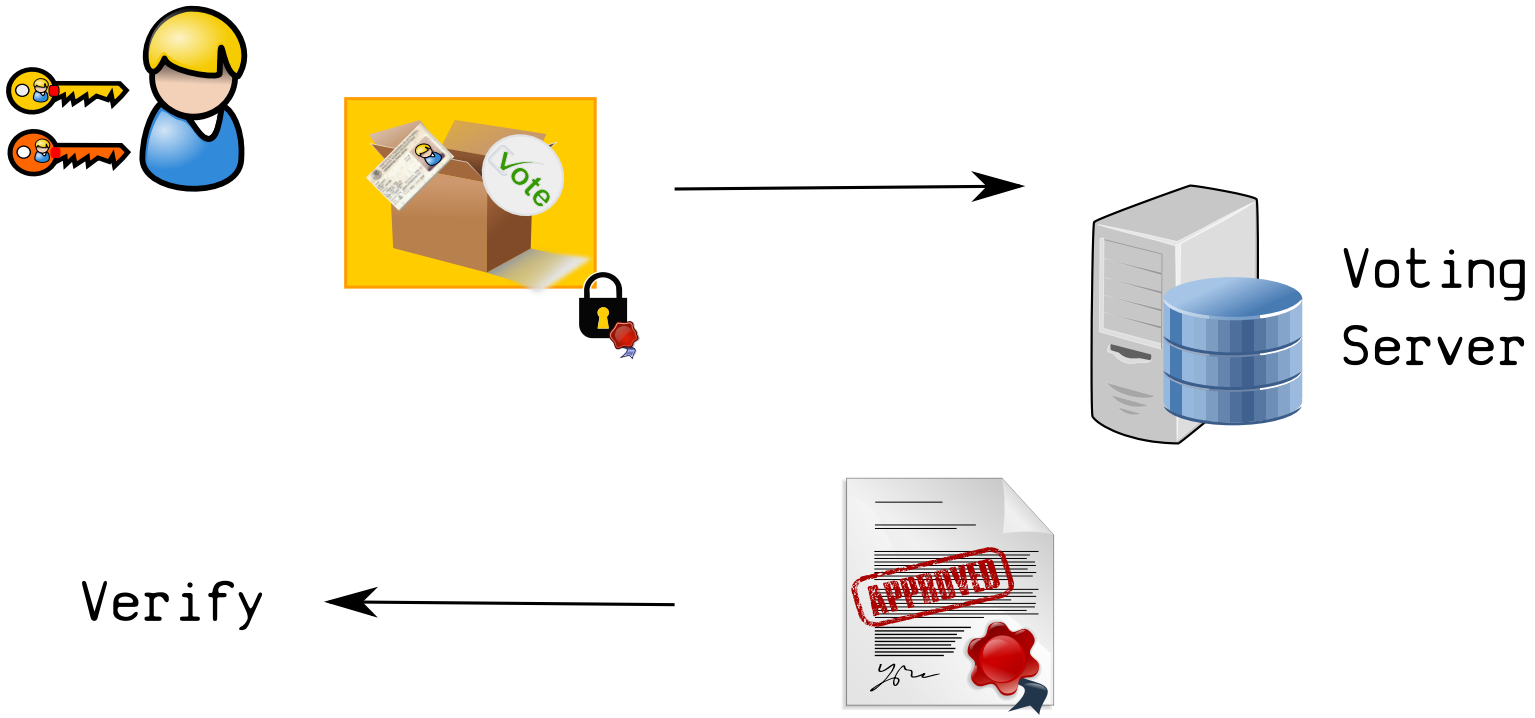
e-voting system based on pairings

An e-voting system based on short and blind signatures by Lopez-Garcia and Rodriguez-Henriquez.



e-voting system based on pairings

With a blind signature, we can cast our vote in the blank ballot.



Conclusions

- The fastest pairing function is not the panacea for some protocols
- Efficient simultaneous pairing implementation is the key to some protocols
- Scalar point multiplication is also a key primitive.
- Protocols with $\mathbb{G}_1 \neq \mathbb{G}_2$ can also be implemented efficiently
- PBC is cheaper than other solutions
- We can do 5 S.M. in $\mathbb{G}_1$, 3 in $\mathbb{G}_2$, and less than 2 expo. in $\mathbb{G}_T$ at approximately the same cost of a pairing

Future work

- More protocol/pairing implementations
- Evaluate the pairing function in a protocol context
- Evaluate the ancillary functions around the pairing
- Pairings with $\mathbb{G}_1 \neq \mathbb{G}_2$ should be encouraged
- Encourage PBC
- Consider multicore environment (ongoing work)
- More optimizations on the ancillary functions.
- Lean modular reduction (a.k.a. Lazy reduction)

Question time

- Thank you for your attention.

Table of Contents

Timings

Using a Intel Core i7 2600K, Sandy Bridge

Operation	Clock cycles
RegularPairing	2108 Kclk
New Pairing	1550 Kclk
G1mul K	232.89Kclk
G1mul U	304.44Kclk
G2mul K	378.26Kclk
G2mul U	535.69Kclk
GTexpo K	617.32Kclk
GTexpo U	931.98Kclk

Cinvestav

Detailed timings

Operation	Clock cycles
G1 Add JJA	1.92Kclk
G1 Add JJJ	2.44Kclk
G1 Dbl A	1.20Kclk
G1 Dbl J	1.44Kclk
G2 Add JJA	5.11Kclk
G2 Add JJJ	6.70Kclk
G2 Dbl A	3.03Kclk
G2 Dbl J	2.92Kclk
GT Sqr	3.78Kclk
GT Mul	9.55Kclk

Timings of Water's CPABE

LSSS ABE Protocole	CPU cycles		
	Theoretical	Expected	Measured
Encrypt	5 922 K	6 105 K	6 378 K
Keygen	1 989 K	2 014 K	2 114 K
Decrypt ($\Delta = 1$)	8 716 K	9 101 K	9 489 K
Decrypt ($\Delta \neq 1$)	9 612 K	10 051 K	10 438 K

Table: Cost of the protocol steps

Timings of the e-voting protocol

Scheme	# Cryptographic operation	# Cycles
Kharchineh & Ettelace	4 RSA-public	6,053,528
	6 RSA-private	253,251,894
	4 DLP-exponentiations	87,135,920
	Total	**346,441,342**
Li et al.	15 RSA-public	22,700,730
	9 RSA-private	379,877,841
	Total	**402,578,571**
Chung & Wu	5 RSA-public	7,566,910
	4 RSA-private	168,834,596
	Total	**176,401,506**
The proposed scheme	1 scalar multiplication in $\mathbb{G}_2$	380,000
	6 scalar multiplications in $\mathbb{G}_1$	1,800,000
	6 map-to-point functions H_1	1,890,000
	8 bilinear pairings	14,630,000
	Total	**18,700,000**

Detailed e-voting system... 1/2

An e-voting system based on short and blind signatures by Lopez-Garcia and Rodriguez-Henriquez.

Voter		Authentication Server (AS)
	Authentication phase	
$b, d_t \in \mathbb{Z}_r$		
$V_t = d_t Q \in \mathbb{G}_2$		
$m = m2s(V_t) \in \{0,1\}^{1016}$		
$\tilde{M} = bH_1(m) \in \mathbb{G}_1$		
$S_{\tilde{M}} = d_V \tilde{M} \in \mathbb{G}_1$		
	$\{ID_V, t, \tilde{M}, S_{\tilde{M}}\}$ $\longrightarrow$	
		$e(Q, S_{\tilde{M}}) \stackrel{?}{=} e(V_V, \tilde{M})$
		$\tilde{S} = d_{AS}\tilde{M} \in \mathbb{G}_1$
	$\{t, \tilde{S}\}$ $\longleftarrow$	
$S_{V_t} = b^{-1}\tilde{S} \in \mathbb{G}_1$		

... detailed e-voting system 2/2

Voting phase

		Voting server (VS)
$S_v = d_t H_1(v) \in \mathbb{G}_1$		
$B = \{V_t, S_{V_t}, v, S_v\}$	$\xrightarrow{\{B\}}$	$m = m2s(V_t)$
		$e(Q, S_{V_t}) \stackrel{?}{=} e(V_{AS}, H_1(m))$
		$e(Q, S_v) \stackrel{?}{=} e(V_t, H_1(v))$
		$a \in \mathbb{Z}_r$
		$ACK = H(V_t \|\| S_{V_t} \|\| v \|\| S_v \|\| a)$
		$S_{ACK} = d_{VS} H_1(ACK)$
	$\xleftarrow{\{ACK, S_{ACK}\}}$	
$e(Q, S_{ACK}) \stackrel{?}{=} e(V_{VS}, H_1(ACK))$		