Introduction
Signature schemes

Luis J. Dominguez Perez

CRYPTO-CO, Julio 1 de 2016

Contenido, sección 1

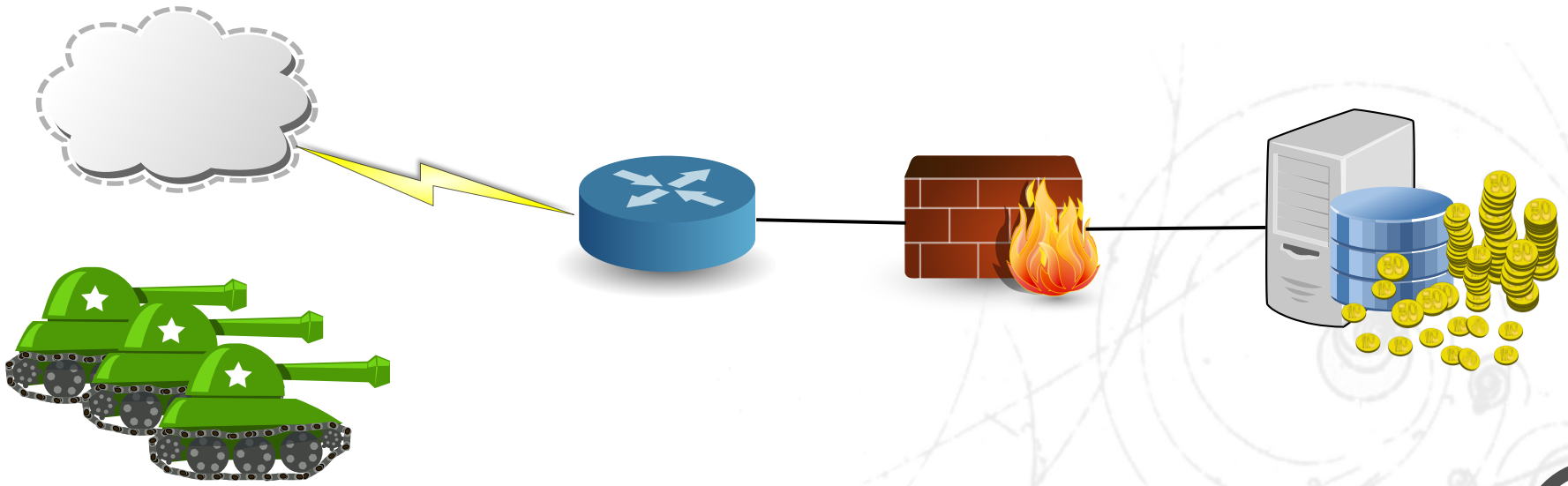
Deployed schemes

- Conventional assumptions about intractability:
  - Integer Factorisation (IFP): RSA
  - Discrete Logarithm (DLP): Diffie-Hellman (DHP), and its variants: ECC y PBC

- These assumptions could be reduced to the Hidden Subgroup Problem (HSP).

Balance between the protection, and the protected

- The main concern about a security system is how long an attacker would need to break it, and how many resources he would be needing.
- The cost to break a system, measured both in time, and invested money in computing resources most be larger than the protected information.



Benefits, and timings

- When an attacker need 100 million USD to get a benefit of 100 mil USD, then, the attacker would desist

- When we say that an attacker would need around 100 years to break a system, this is computed at the start of the attack
  - As time goes on, new methods, and better resources become available
  - An attacker could be updating its resources, y reduce drastically the initially estimated 100 years

Cost estimation

- In a symmetric scheme, the time to live of the data protection could be estimated on the time it takes to break the scheme by brute force, using a dictionary with possible keys, or another method

- In "Using the cloud to determine key strengths", Kleinjung et al. proposed an estimation scheme based on the cost of breaking a cryptographic scheme with the use of the Amazon cloud servers.
  - By the time, they estimated that breaking a 128-bits security sysmtem using the Amazon services would cost about 10^{27} USD.

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

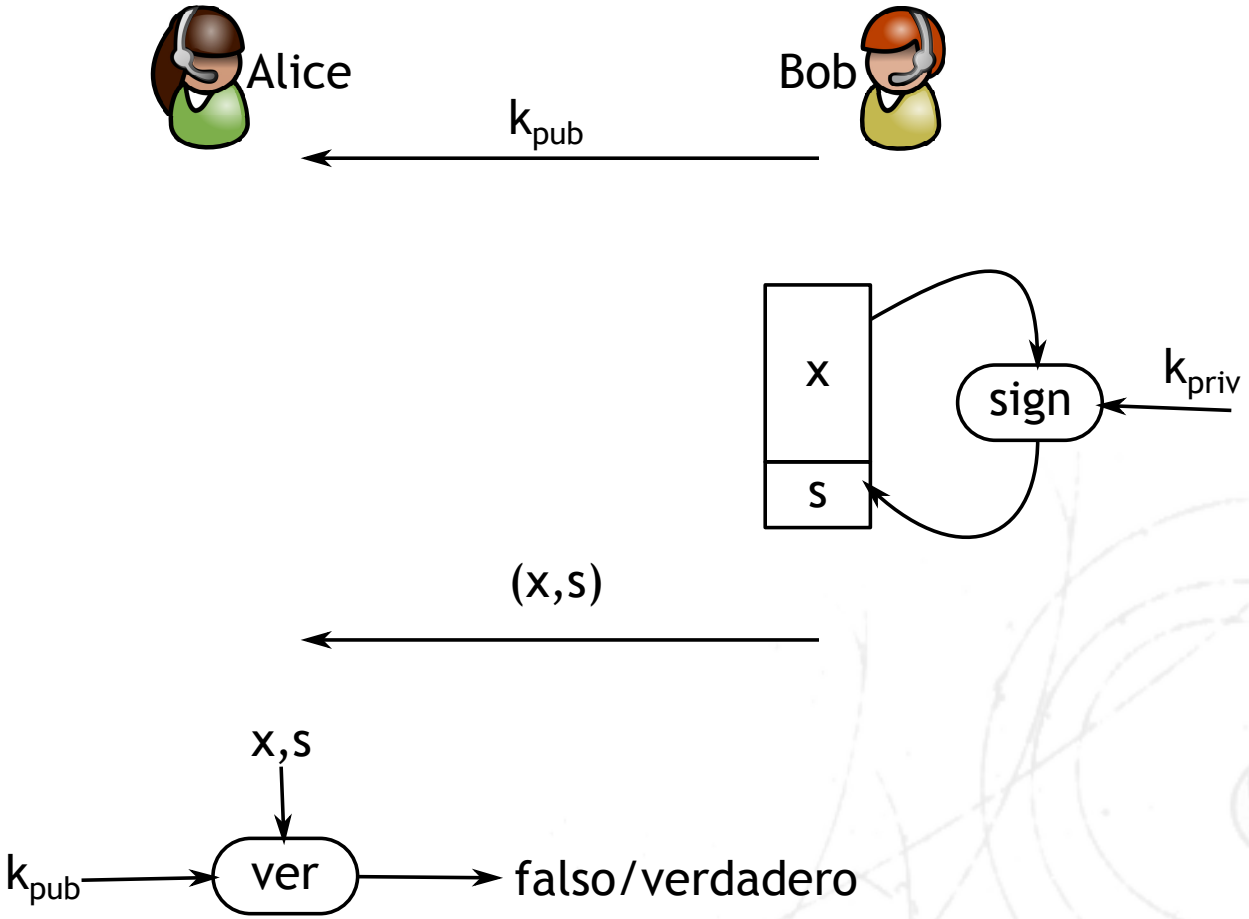
Contenido, sección 2

Digital Signatures

- Demonstrating that a certain person generated a given message is a critical application.
- In the "analogic" world, we tend to use hand signatures on paper.
- Only the person who signed can reproduce it again

Scheme

This is possible using public key cryptography. The signatory signs a message with his private key, the recipient uses his public key to verify the message.



Contenido, sección 3

RSA

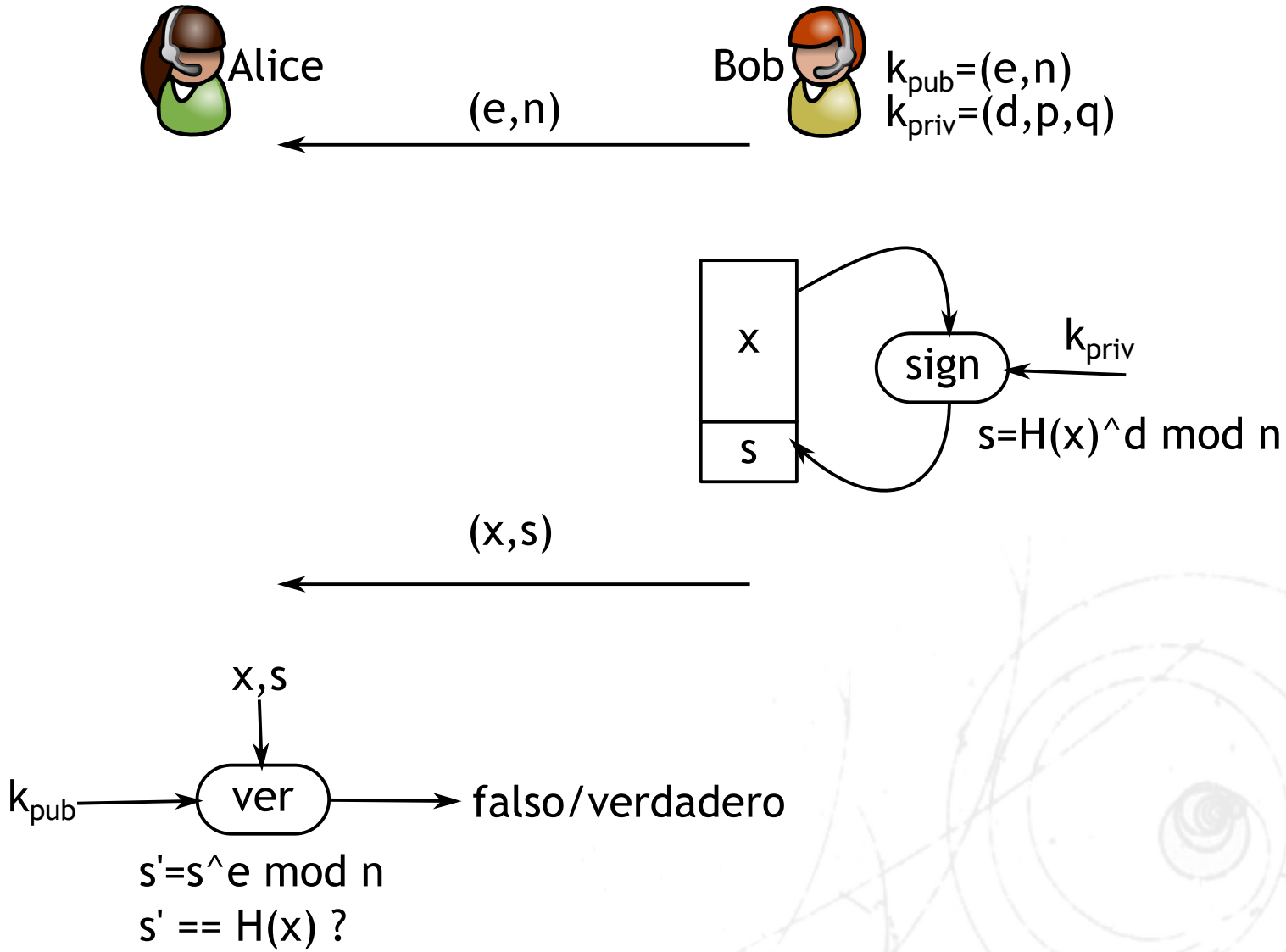
Designed in 1977 by Ron Rivest, Adi Shamir, and Leonar Adleman.

- Let p y q two different, and large random prime numbers
- The module n is the product of p, and q
- The function $\Phi(n) = (p-1)(q-1)$
- Choose $1 < e < \Phi(n)$, such that GCD$(e, \Phi(n)) = 1$; typically $e = 2^{16} + 1$ (but there's a new recommendation)
- Compute $d \equiv e^{-1} \bmod \Phi(n)$

The public key is (e, n). The private key is (d, p, q).

RSA Signature

Basic RSA signature



Contenido, sección 4

ElGamal

- The Elgamal encryption was proposed by Taher Elgamal en 1985.
- This is an extension of the Diffie-Hellman Key Exchange (DHKE)

Elgamal signature

- Key generation:
  - Choose a prime p
  - Get a generator $\alpha \in \mathbb{Z}_p^\star$
  - Get a random number d, with $2 < d < p - 2$
  - Compute $\beta = \alpha^d \bmod p$

Elgamal signature of a message

- Sign message:
  - Given a message M
  - Choose an Ephemeral k_E, with $2 < k_E < p - 2$, and $\text{GCD}(k_E, p-1) = 1$
  - Compute $r \equiv a^{k_E} \bmod p$
  - Compute $s \equiv (M - d \cdot r) k_E^{-1} \bmod p - 1$

- The signature of M is (r, s)

Elgamal signature verification

- Signature verification:
  - Compute $t \equiv \beta^r \cdot r^s \bmod p$

- If $t \equiv \alpha^M \bmod p$, the signature verifies.

Example: message M to sign

$$p = 29,\ \alpha = 2$$
$$d = 12$$
$$\beta = \alpha^d \equiv 7$$

$$\xleftarrow{k_{\text{pub}}(p,\alpha,\beta)=(29,2,7)}$$

$$k_E = 5$$
$$(5, 28) = 1$$
$$x = 26$$
$$r = 2^5 \equiv 3$$
$$s = -10 \cdot 7 \equiv 26$$

$$\xleftarrow{(26,(3,26))}$$

$$t = 7^3 \cdot 3^{26} \equiv 22$$
$$\alpha^x \equiv 2^{26} \equiv 22$$
$$t \equiv \alpha^x \Rightarrow \text{OK}$$

Contenido, sección 5

DSA Signature

The standard signature DSA has the following steps:

- Key Generation:
  - Find a prime p, with $2^{1023} < p < 2^{1024}$
  - Find a prime q, with $2^{159} < q < 2^{160}$
  - Find a generator α in $\mathbb{F}_q$
  - Choose a random number d, with $1 < d < q$
  - Compute $\beta = \alpha^d \bmod p$

- The keys are:
  - Public: (p, q, α, β)
  - Private: d

DSA message signing

- Message signature:
  - Given a message M
  - Choose an ephemeral k_E, with $0 < k_E < q$
  - Compute $r \equiv (a^{k_E} \bmod p) \bmod q$
  - Compute $s \equiv (SHA(M) + d \cdot r)k_E^{-1} \bmod q$

- The signature of M is (r, s)

DSA signature verification

- Signature Verification:
  - Compute $w \equiv s^{-1} \bmod q$
  - Compute $u_1 \equiv w \cdot SHA(M) \bmod q$
  - Compute $u_2 \equiv w \cdot r \bmod q$
  - Compute $v \equiv (\alpha^{u_1} \cdot \beta^{u_2} \bmod p) \bmod q$

- If $v \equiv r \bmod q$, the signature verifies.

Example: message M to sign

$$p = 59,\ q = 29$$
$$\alpha = 3,\ d = 7$$
$$\beta = \alpha^d \equiv 4$$

$$\xleftarrow{k_{\text{pub}}(p,q,\alpha,\beta)=(59,29,3,4)}$$

$$k_E = 10$$
$$r = (3^{10} \bmod 59)$$
$$\equiv 20 \bmod 29$$
$$s = (26 + 7 \cdot 20) \cdot 3$$
$$\equiv 5 \bmod 29$$

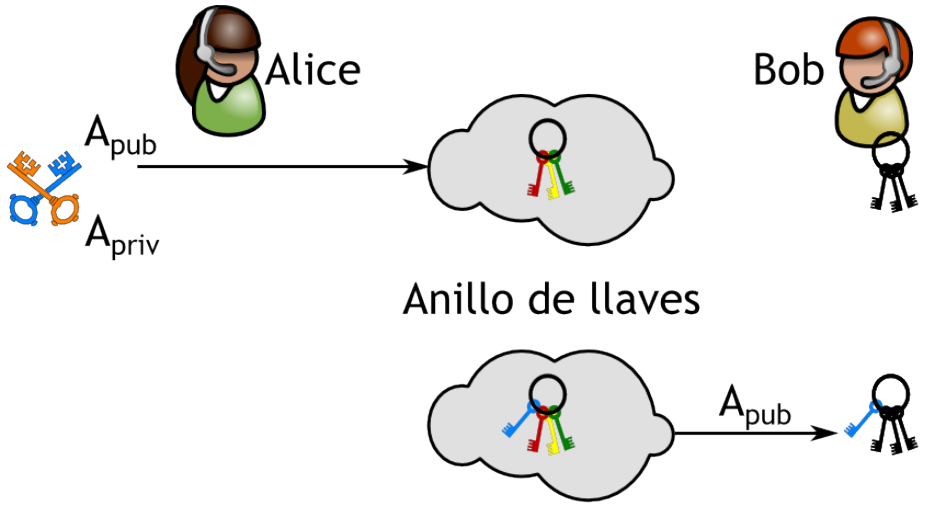
$$\xleftarrow{(M,(r,s))}$$

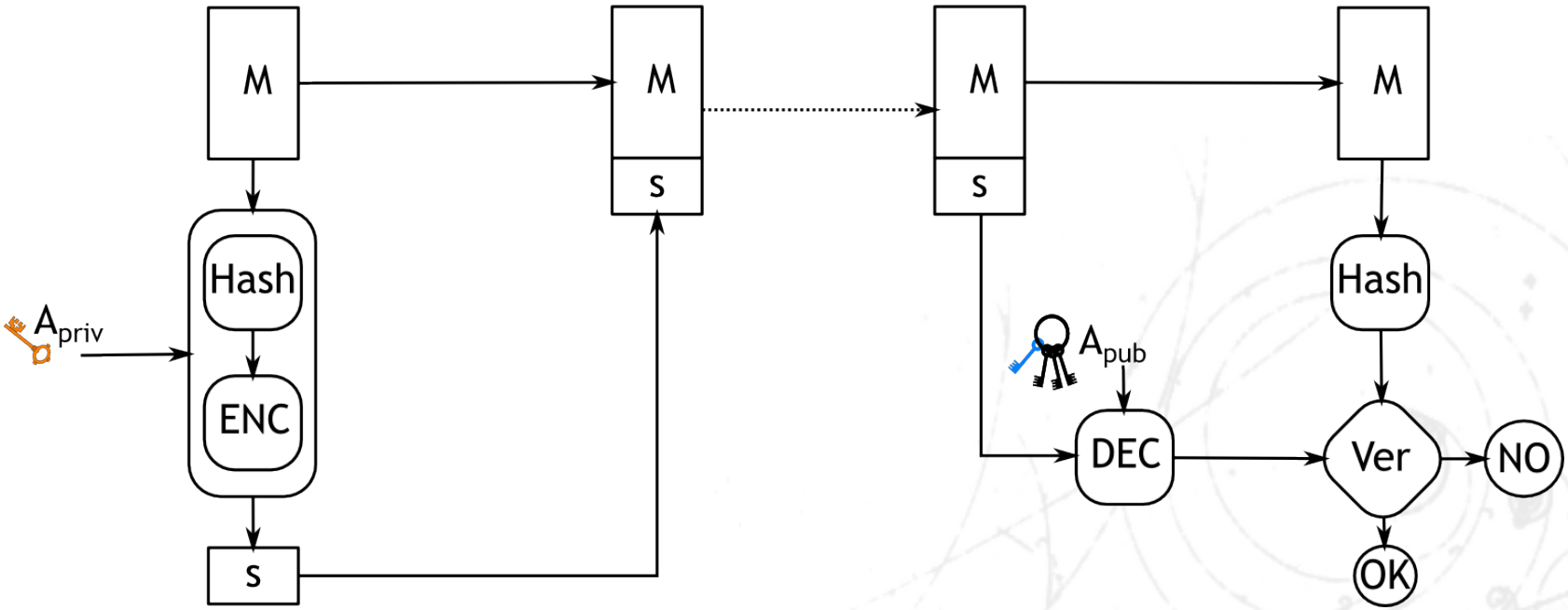
$$w = 5^{-1} \equiv 6 \bmod 29$$
$$u_1 = 6 \cdot 26 \equiv 11 \bmod 29$$
$$u_2 = 6 \cdot 20 \equiv 4 \bmod 29$$
$$v = 20 \equiv (3^{11} \cdot 4^4 \bmod 59) \bmod 29$$
$$v \equiv r \bmod 29 \Rightarrow \text{OK}$$

Other signatures

- A particularly efficient signing method for applications in which the power usage, or the space is restricted are the short signatures based on elliptic curves.
- The Elliptic Curve Based Cryptography is a type of public key cryptography that requires less space to store the key than other similar schemes.
- The Elliptic Curve Cryptography is much more elaborated, but it permits the efficient implementation of interesting security protocols, or at a lower cost.

Digital Signature





Contenido, sección 6

Stuxnet

Originally of unknown origin, this was a Windows worm that attacked Siemens equipment in orer to spy, and destroy industries.

It used to attack the industry in Iran sometime after some rumors that they were working with Uranium, which could be used for military purposes.

http://pastebin.com/a1DeRyFN

Stuxnet

One year after the initial attack, about 1,000 centrifugal machines in Natanz electric installation were failing, or broken.

The worm modified the spin operation from too fast to slow, and viceversa, without leaving trace in the logs.

Actually, this worm is known as a prototype between some government agencies of Israel, and the USA that went wild accidentally.

Infection

The worm affected Windows computers with the Siemens software installed.

Then, it attacked the PLC systems in which the stored configuration file included the option to change the spin speed of the turbines.

In some memory blocks it changed the speed at random times, and it installed a rootkit to hide itself, and for the records to be omitted.

Duqu

In 2011, an identical worm were released, but as a spyware to catch key strokes, and other private information.

Both worms were probably created in 2007. Some other variants are there on the internet.

Flame

It attacked Windows computers, it was used for cyber espionage in mid-east countries.

The program taked screenshots, the key strokes, network activity, it could spy Skype conversations, and even use the bluetooth connection to lookup around the computer for close devices.

At some stage, it was ordered to self-destruct, but some copies were captured.

Installation

Basically, the Flame worm used a vulnerability in the Windows services from certain clients, and abused of a configuration error from the Microsoft Servers, letting an attacker to fake a Windows update.

Microsoft Security Advisory (2718704)

Some digital ertificates could permit spoofing.

The active attacks used non-authorized digital certificates, but derived from a Microsoft's Certificate Authority. A non-authorized certificate could be used to fake content, execute phishing attacks, or execute man-in-the-middle attacks. This problem affected to every single Microsoft Windows version.

About the advisory

Essentially, the update revoked a pair of certificates used by the Terminal Server License service

The Terminal services authorizes a computer attached to the Windows network to use computing resources from the server: user interface, Office, etc.

This way, we can have dumb computers inside old, or slow computers, and for sharing software licenses.

Why the revoke?

These certificates were designed to authorize the terminal clients; however, they were linked to the Microsoft's root certificate for signing, and other actions, which included signing Microsoft updates.

Essentially, anyone with a Terminal Services License could not only sign certificates for your old computers, but also they had the code signing option activated (hen, they could sign Windows updates).

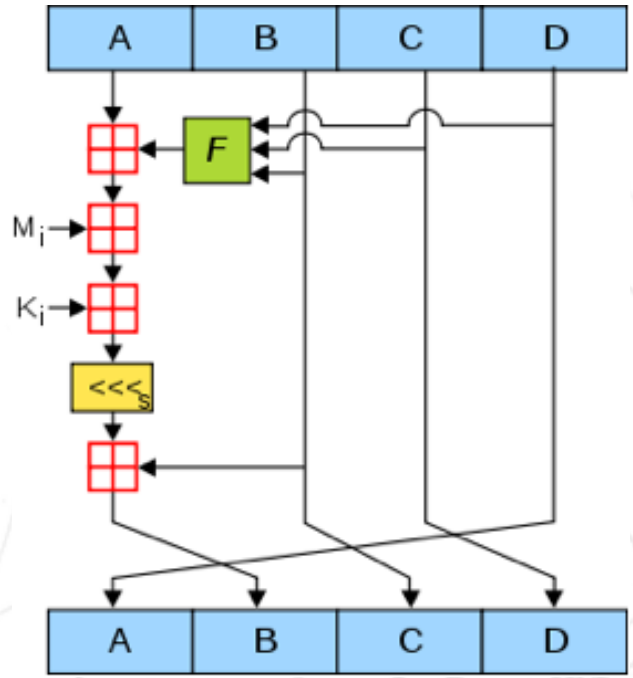
Update about the advisory 2718704

The Flame malware used a cryptographic collision attack in combination to the Terminal Server License Certificates in order to sign malicious code and make it seem to be Microsoft code. However, the code signing without the collision is also possible in older Windows computers

The collision attack was posible, mainly because the 2009 certificate used by the Terminal service was still using an old MD5 hash!

MD5 collision attacks

1996. ”The present attack does nor present practical applications over MD5, but they are clearly soon in the future, MD5 should not be implemented when a collision resistant hash function is required.”



¿Porqué la colisión?

Los certificados de las licencias del cliente de terminal funcionaban para firmar código para las versiones de Windows Vista o anteriores.

Las versiones actuales de Window se quejaban de las extensiones del certificado del servidor de Terminal, por lo que crearon un certificado con una colisión MD5, ¡pero sin las extensiones! //con eso, podían validar malware como software Microsoft en Windows 7.

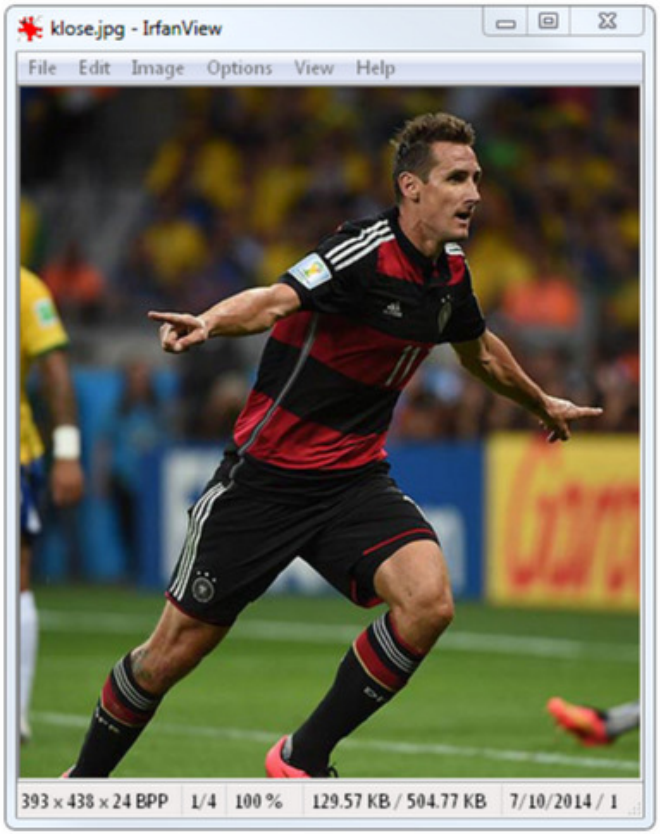
El costo de la colisión

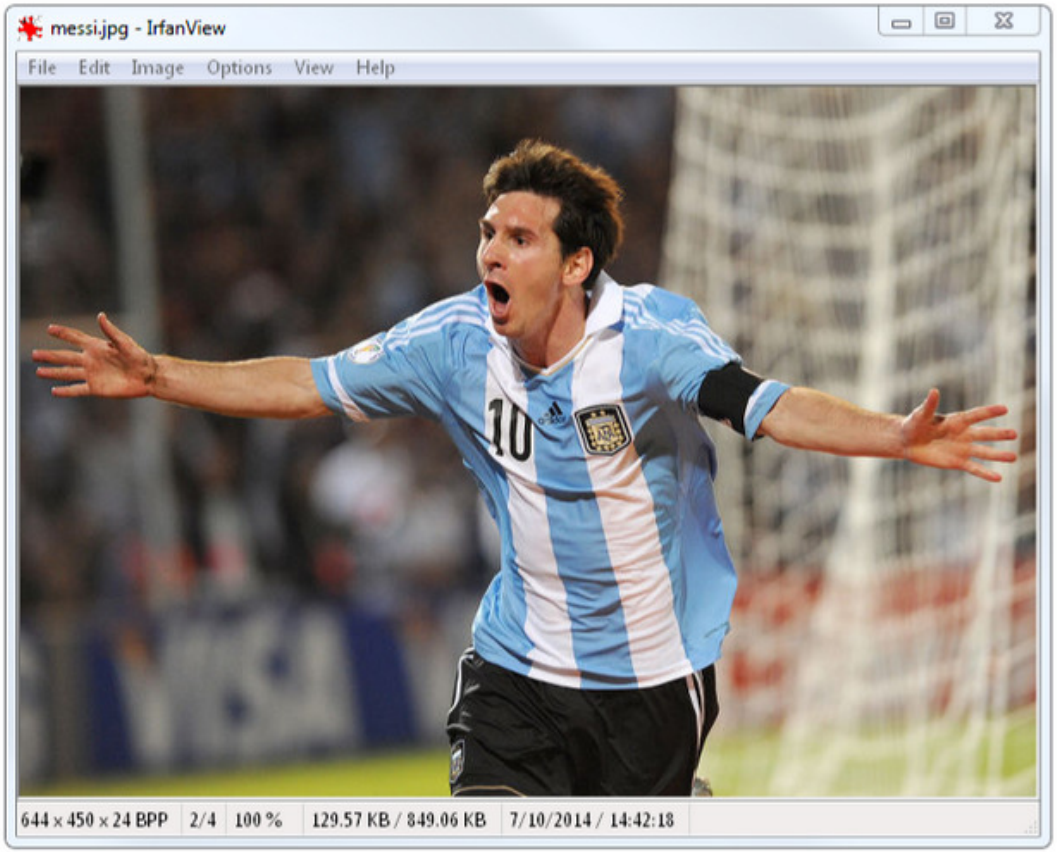
En el 2008, un grupo de científicos encontró una colisión en 1 día utilizando un clúster de PS3.

Cualquier agencia gubernamental con fondos suficientes puede hacer un certificado con una colisión…¿pero quién?

Ahora, el resto de los certificados Microsoft expirarán en el 2020…¡tal vez sea redituable intentarlo!

Colisión SHA-1






```
>crypto_hash *.jpg
fbd1847ac1342acb9c52c30f4b477997938a4a0a *klose.jpg
fbd1847ac1342acb9c52c30f4b477997938a4a0a *messi.jpg
```

Detalles

Para mayores detalles, nos cambiaremos a la presentación de Alex Sotirov's.

Analizando la colisión MD5 en el gusano Flame

También vea: http://blog.didierstevens.com/2012/06/04/flame-before-and-after-kb2718704/

¿Porqué está cambiando la seguridad en las TI?

Basicamente, el gusano Flame estuvo cerca de dos años en la red, ¡sin que nadie lo detectara!

Una vez detectado, se autodestruyó para evitar un análisis forense, sin embargo, algunas firmas de seguridad consiguieron obtener una copia.

¡Se necesitan más maneras para detectar malware!

Contenido, sección 7

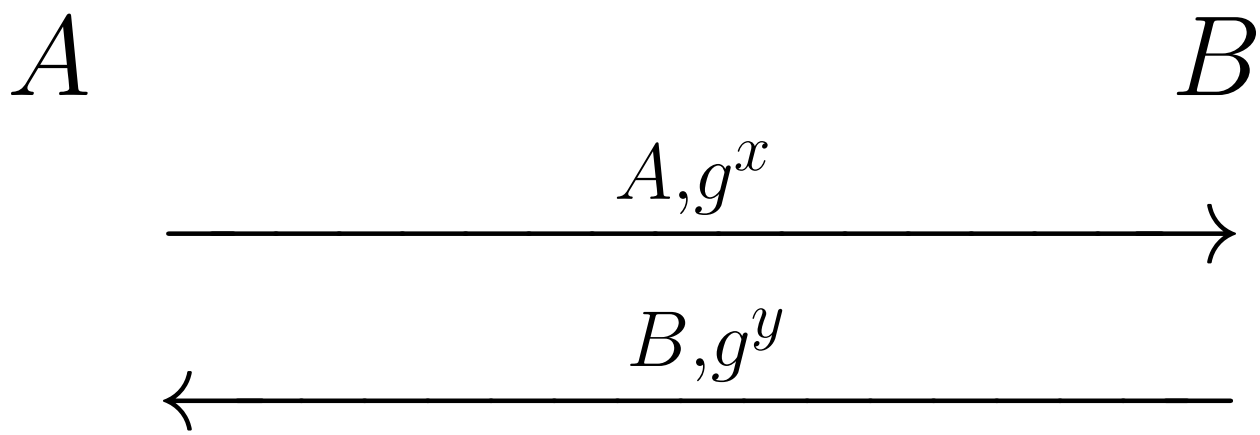
IPSec

- Seguridad de transporte en la capa IP
- Provee tráfico seguro entre dos sistemas IP
- Ofrece servicios de seguridad para los paquetes IP
- Generación y mantenimiento de la Asociación de Seguridad
- Independiente de la aplicación (software)

Para establecer un canal, primero hay que intercambiar claves (simétricas). . .

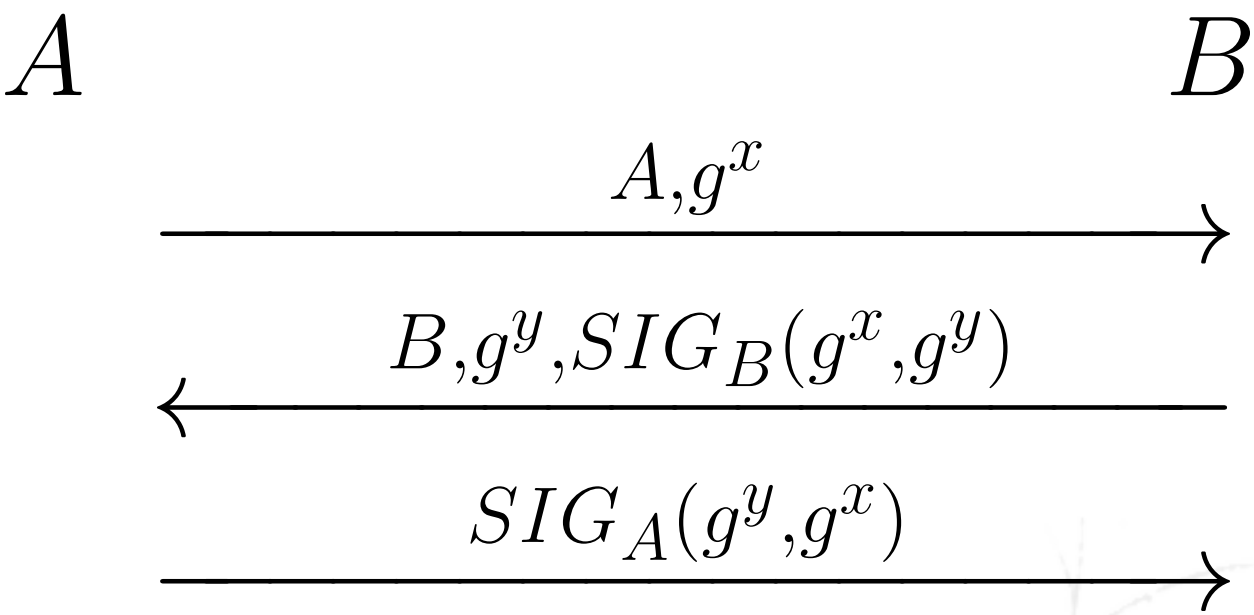
Diffie-Hellman

DH'76 - Diffie-Hellman Exchange

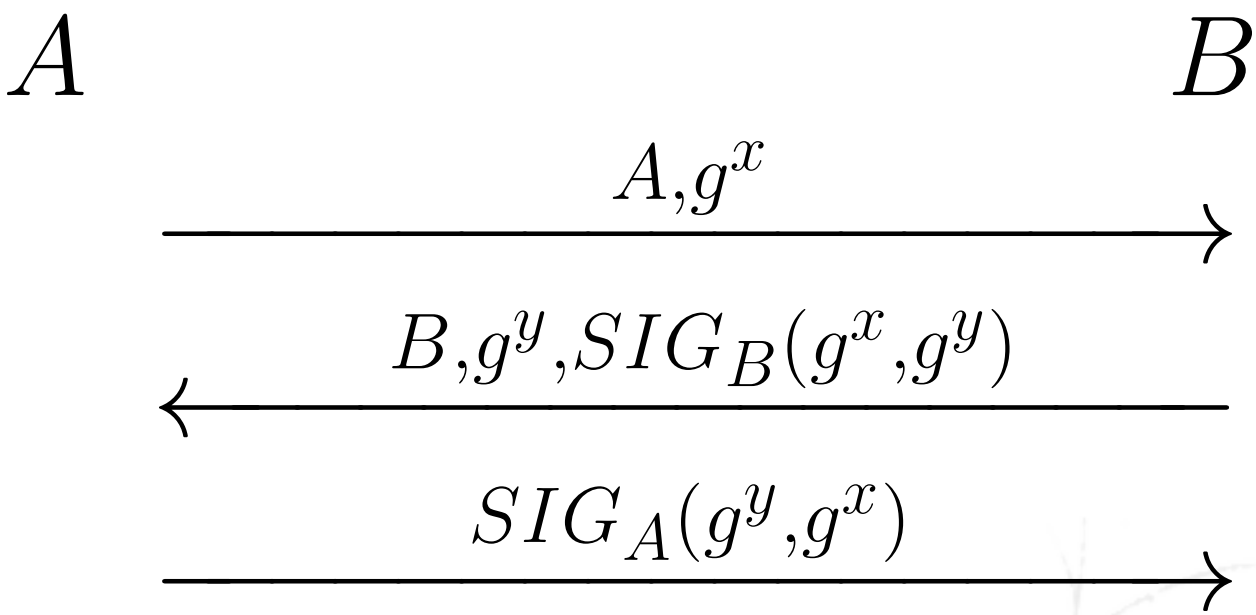


- Ambas partes pueden calcular la llave secreta $K = g^{xy}$
- Dados g^x, y g^y, g^{xy} parece un elemento aleatorio
- Abierto a un ataque M-I-T-M en un ambiente sin autenticación

DH Autenticado básico (BADH)

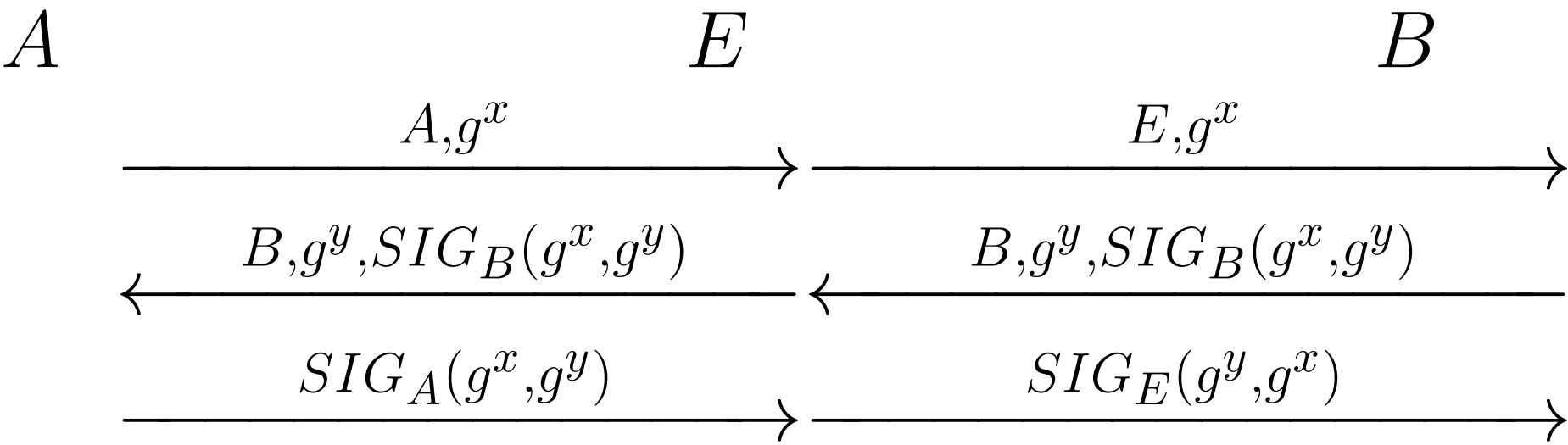


DH Autenticado básico (BADH)

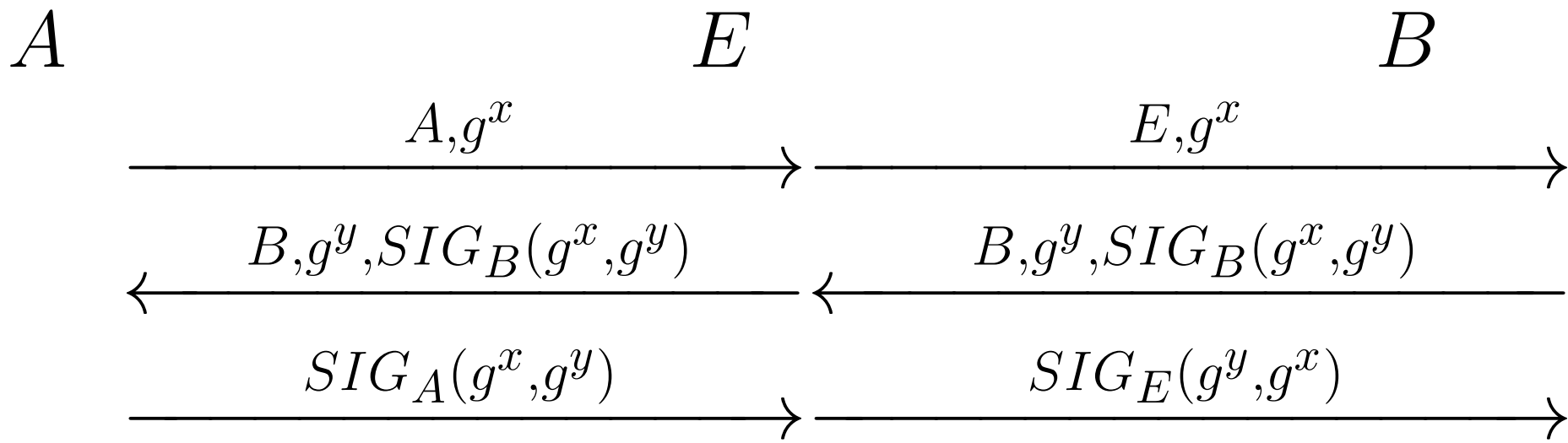


sin embargo...

Ataque a BADH



Ataque a BADH

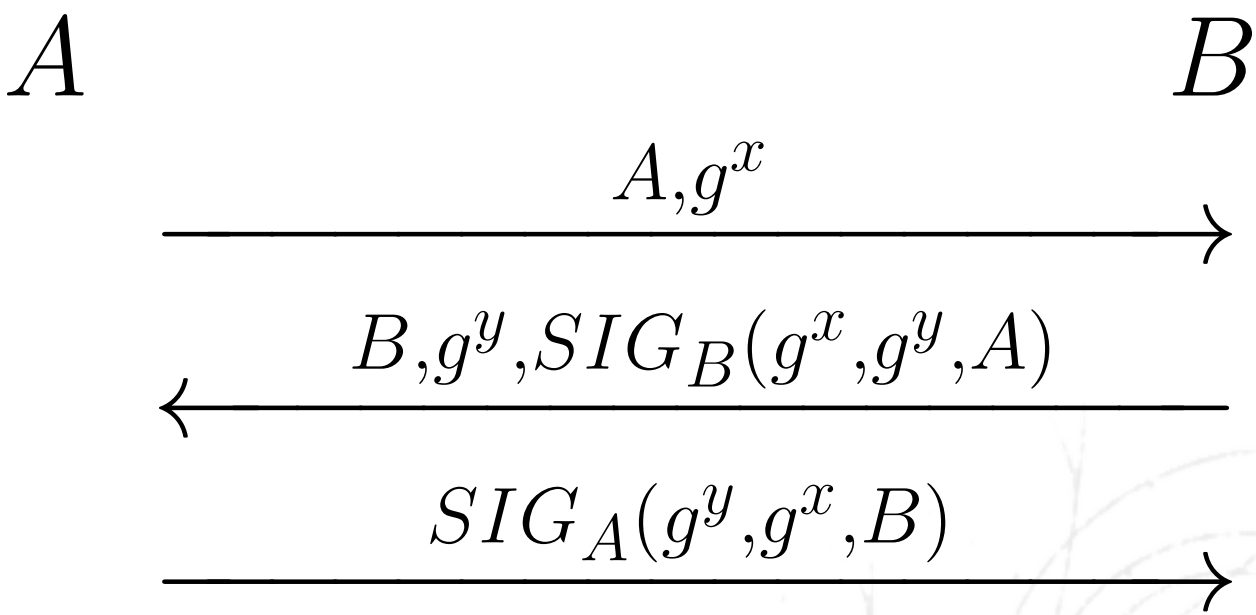


Aunque E no conoce $K = g^{xy}$, B recibe lo que le envíe E pensando que es A.

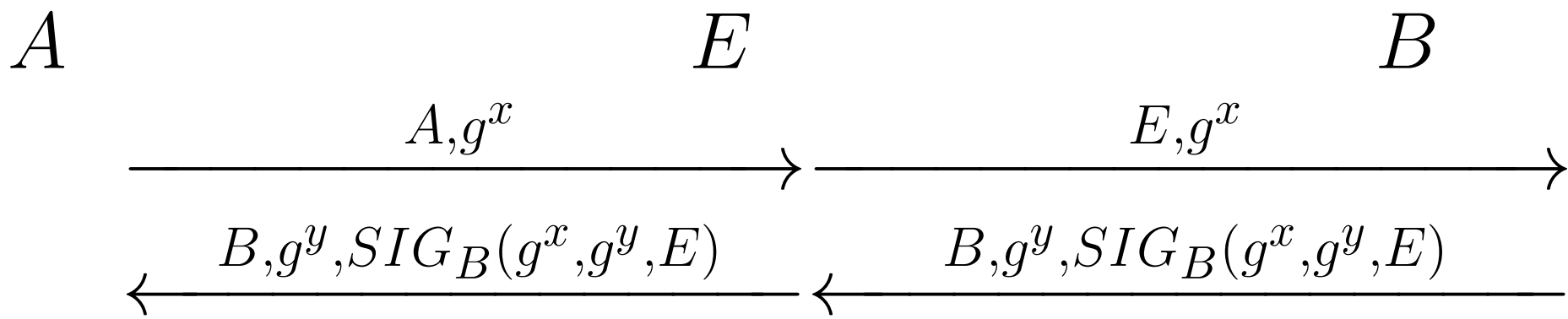
- A recibe mensajes de B correctos, E no puede hacer nada.
- B recibe mensajes de A con identificador de E.

Solución ISO-9796. . .

Incluir la identidad del receptor dentro de la firma:



Ataque. . .



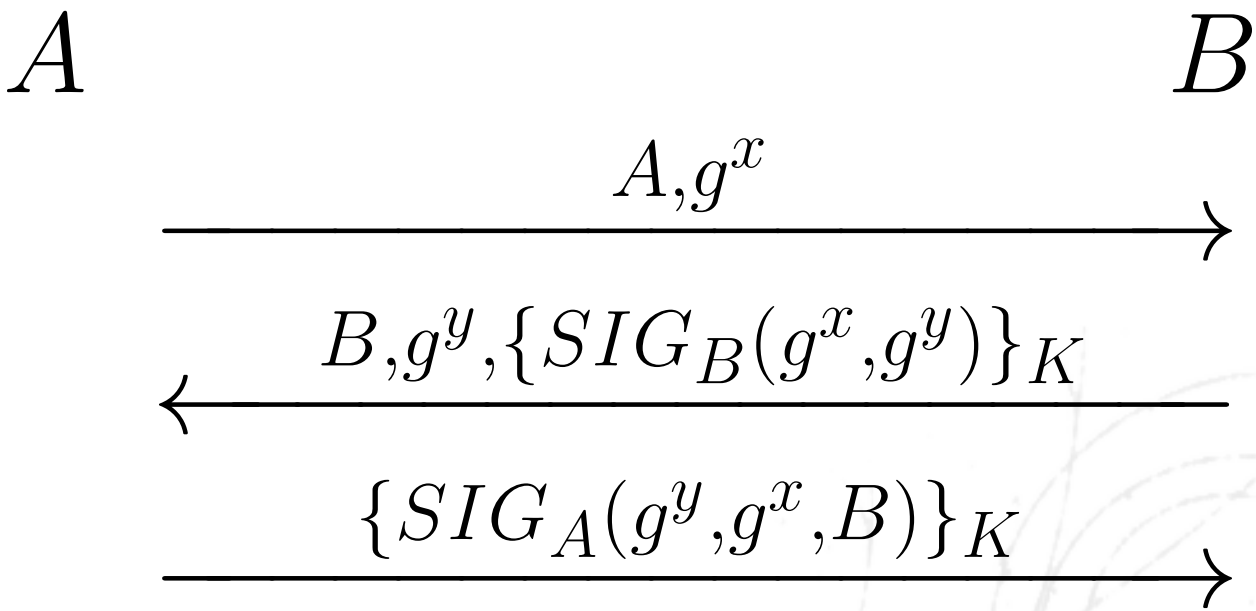
- A: ¡Ajá! B se está comunicando con E, no conmigo
- E no puede producir $SIG_B(g^x, g^y, A)$

¿Es seguro?

- Es técnicamente seguro
- No sirve para protección de identidades:
  - B necesita saber la identidad de A antes de autenticarse, lo mismo para A
  - Privacidad: hay evidencia firmada de la comunicación
  - Hacer que cada parte firme su identidad resuelve el problema de la privacidad, pero volvería el protocolo inseguro (ataque M-I-T-M).

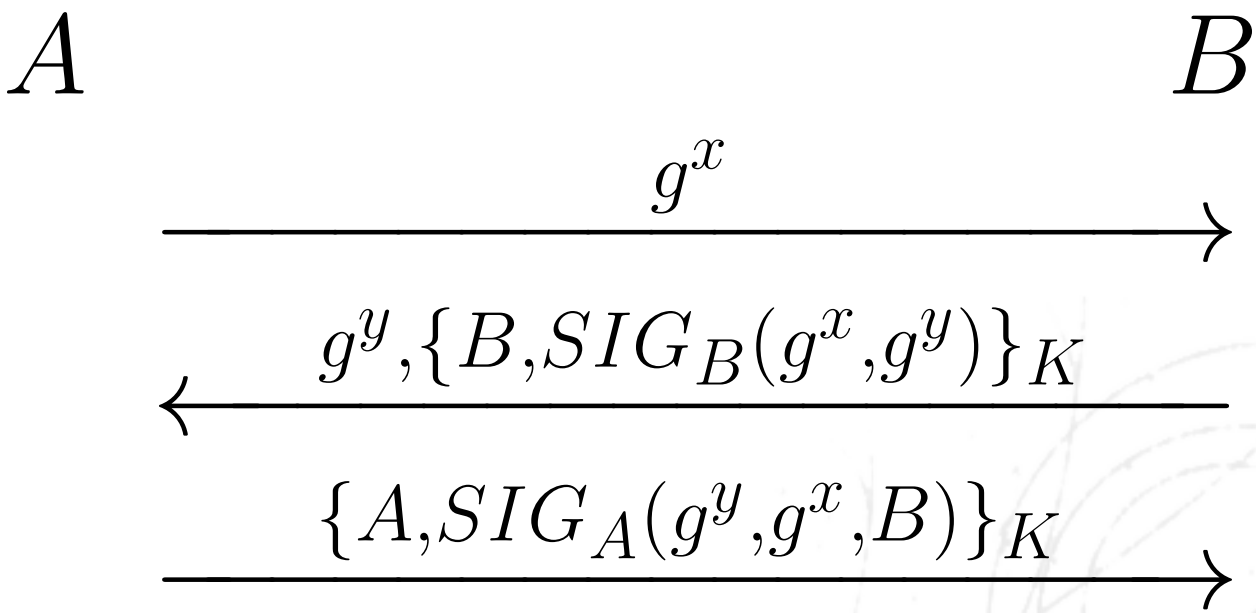
Otra solución: STS

Probar conocimiento de la $K = g^{xy}$ utilizando un cifrado simétrico:



¿Es seguro?

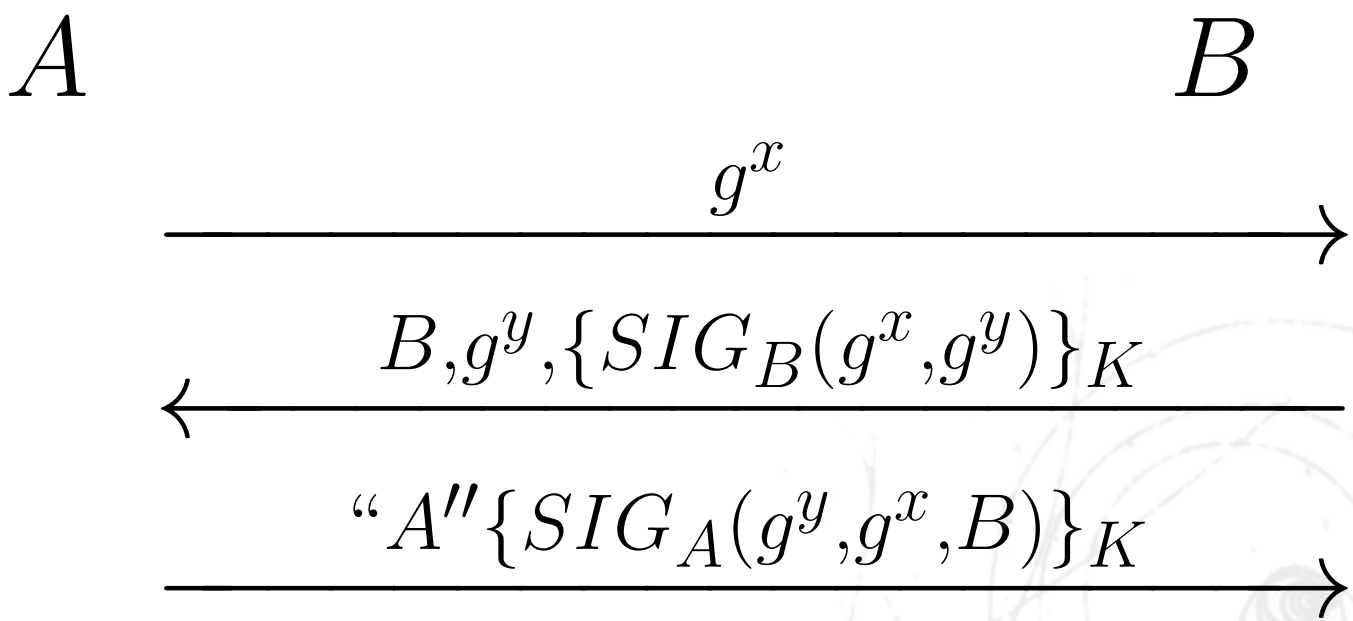
- Es seguro
- se puede extender para proteger a las identidades de la transacción



. . .¿es seguro?

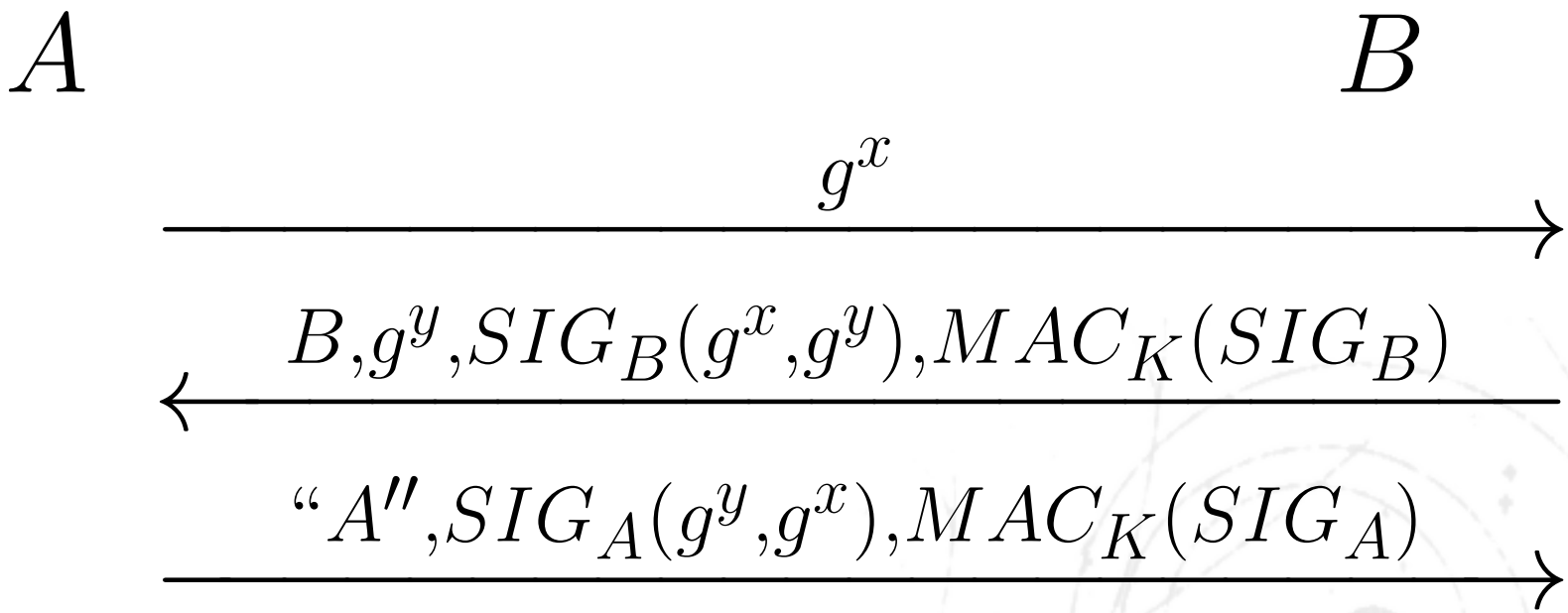
El cifrado, no es la función correcta para provar conocimiento de una llave

- alguien podría registrar como suya la llave pública de otra persona, y montar un ataque I-M (y sin conocer $K = g^{xy}$)



STS con MAC

Algunos esquemas verifican la identididad, pero no la posesión de la llave privada (PoP)

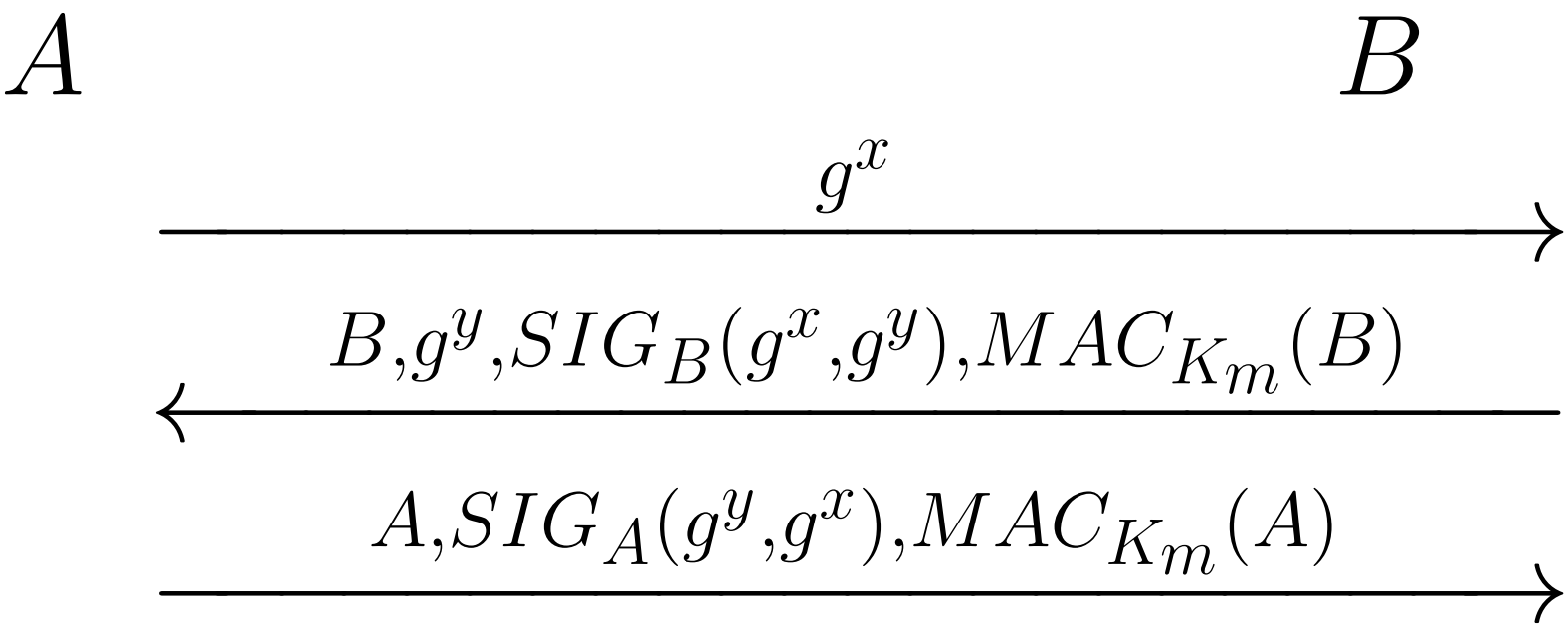


¿Qué pasa?

- El punto es que probar que se conozca la $K = g^{xy}$ no es el problema.
- Lo que se requiere es:
  - asociar la llave K con las identidades

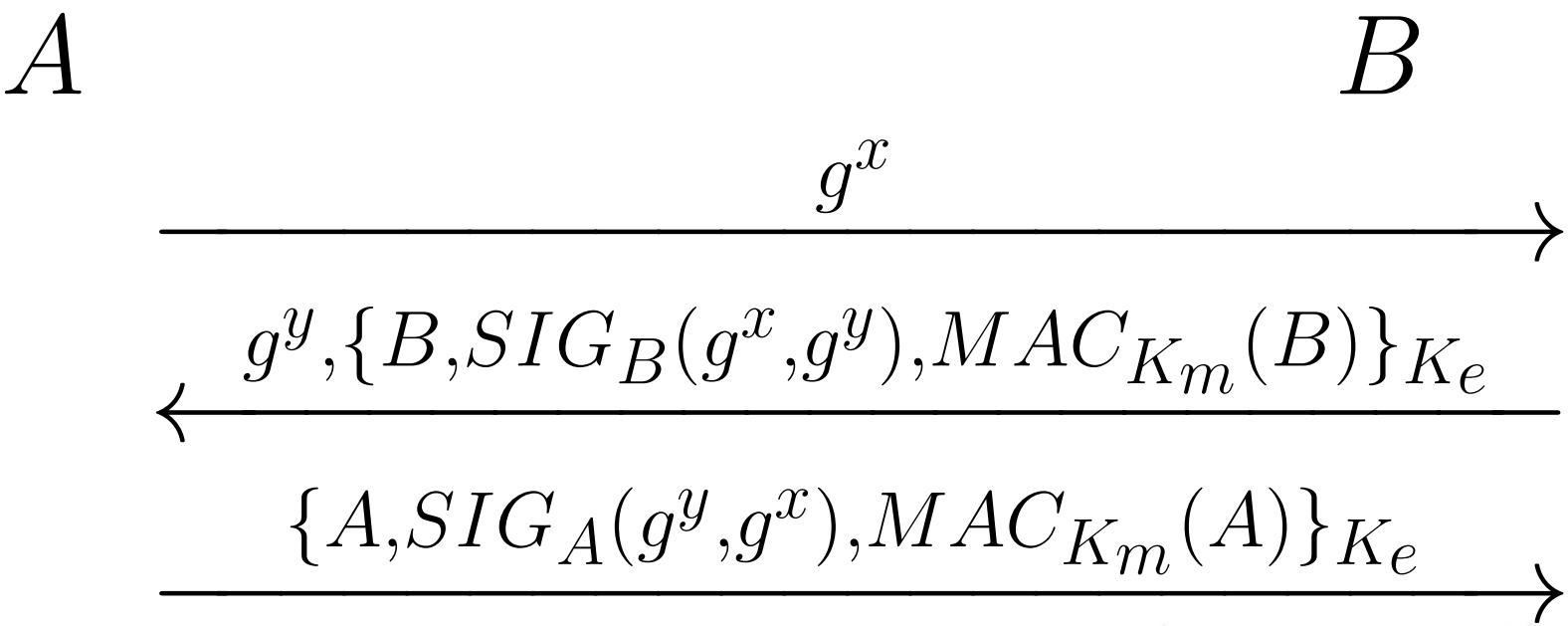
- La solución sería:
  - SIGn-and-MAC su identidad

SIGMA básico



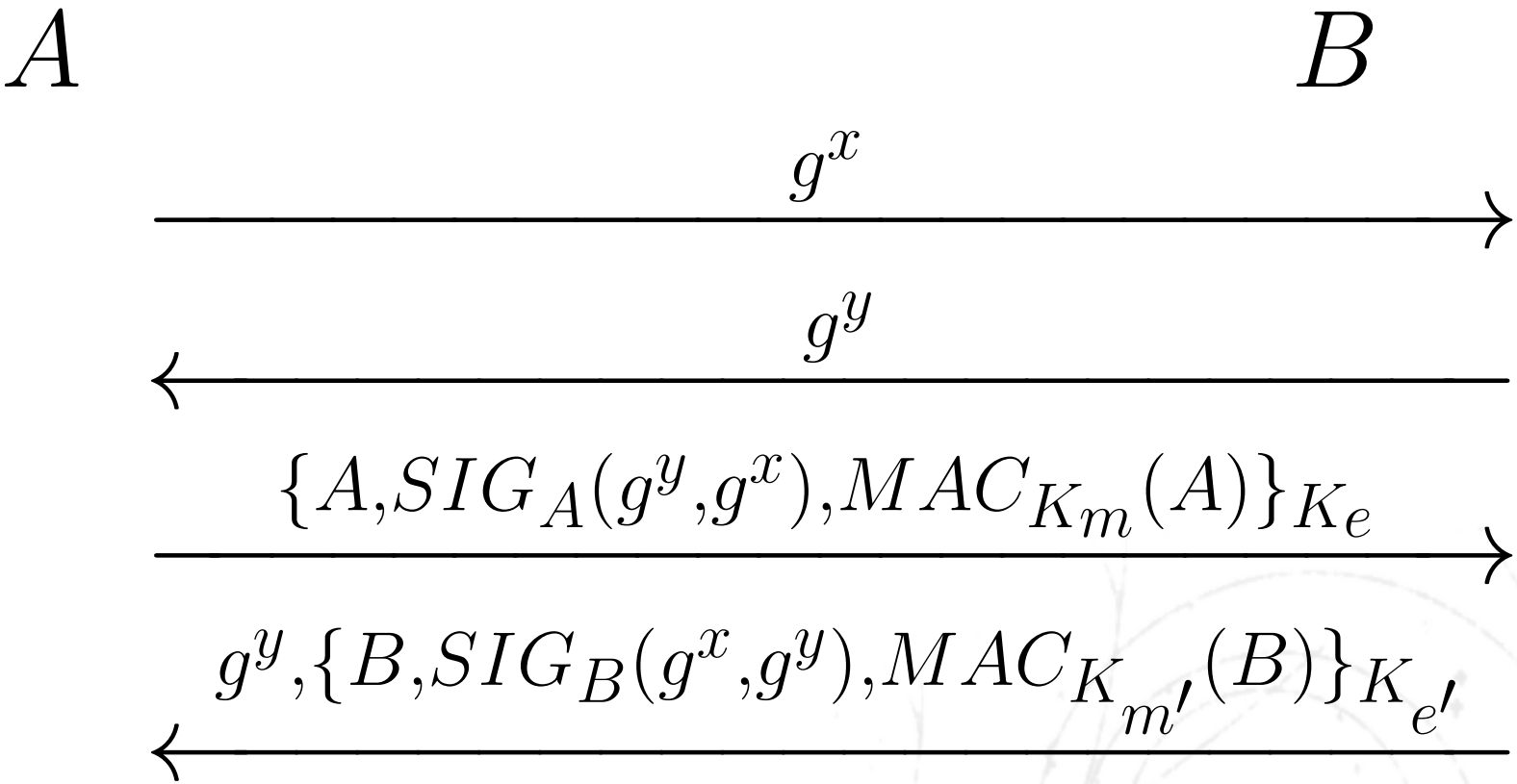
- Se genera una llave K_m a partir de la llave K
- SIG y MAC tienen roles complementarios un ataque M-I-T-M y para asociar la identidad
- No requiere conocer el ID de las partes para su identificación.

SIGMA-I: protección activa del ID del Iniciador



- Se genera una llave K_m, y una K_e a partir de la llave K
- B es el primero que se identifica
- La identidad del Iniciador (A) está protegida.

SIGMA-R: protección activa del ID del Destinatario (R)



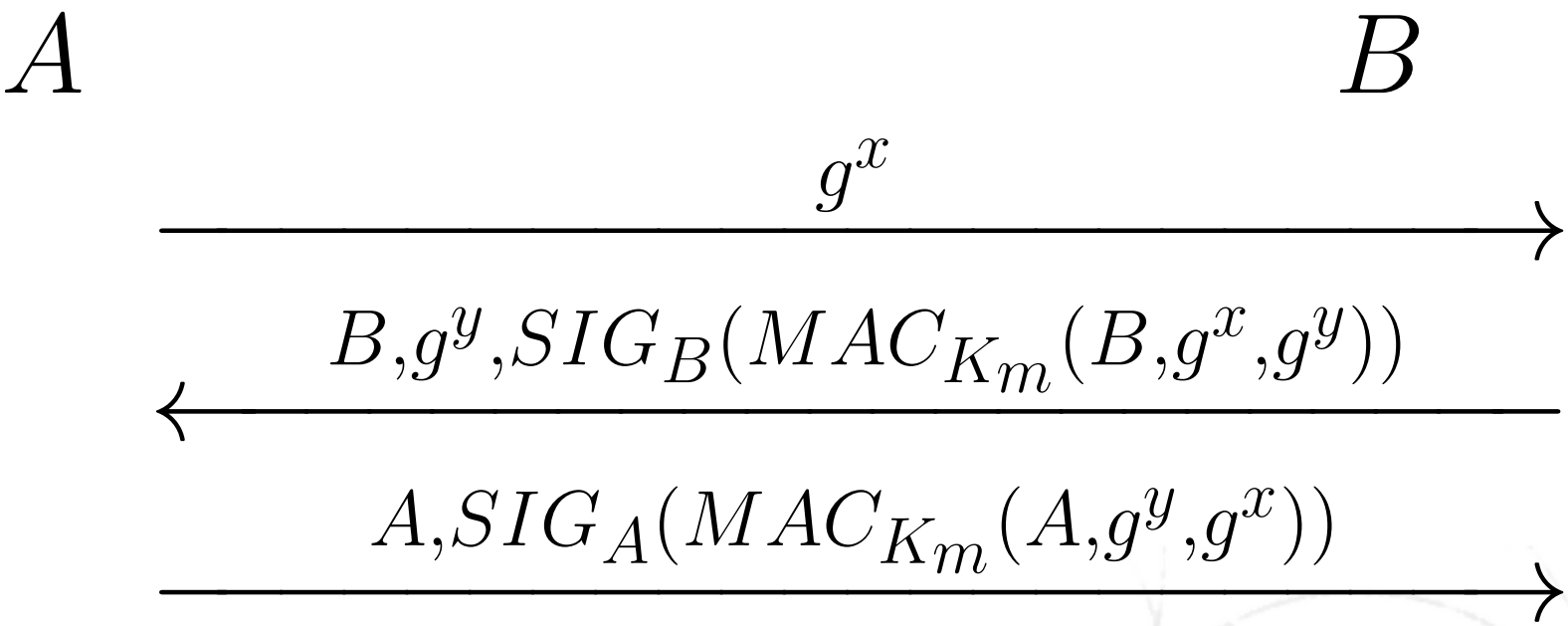
- Se generan llaves K_m, $K_{m'}$, K_e, y K'_e a partir de la llave K

IKE: Internet Key Exchange

- Crea SAs (Asociaciones de Seguridad) para utilizarse en un enlace IPSec
- Administra las SAs

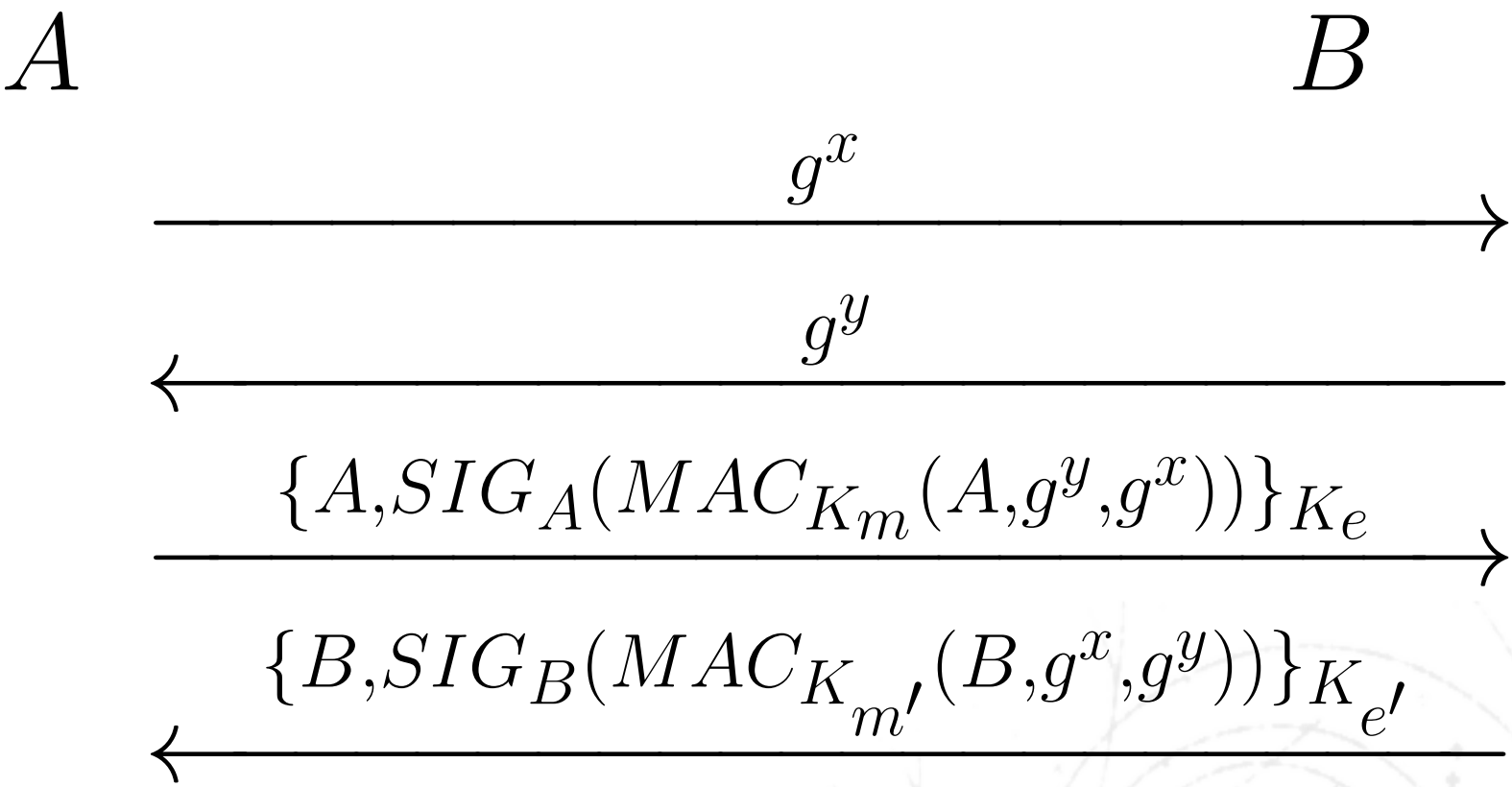
- Cuando un nodo A quiere comunicarse con B bajo la protección de un enlace IPSec, es necesario un intercambio de llave simétrica.
- La derivamos de SIGMA, visto recientemente.

Variante de IKEv1: MAC bajo SIG



- IKE modo “agresivo” (sin protección de ID).

IKE modo principal



- Se generan llaves K_m, $K_{m'}$, K_e, y K'_e a partir de la llave K