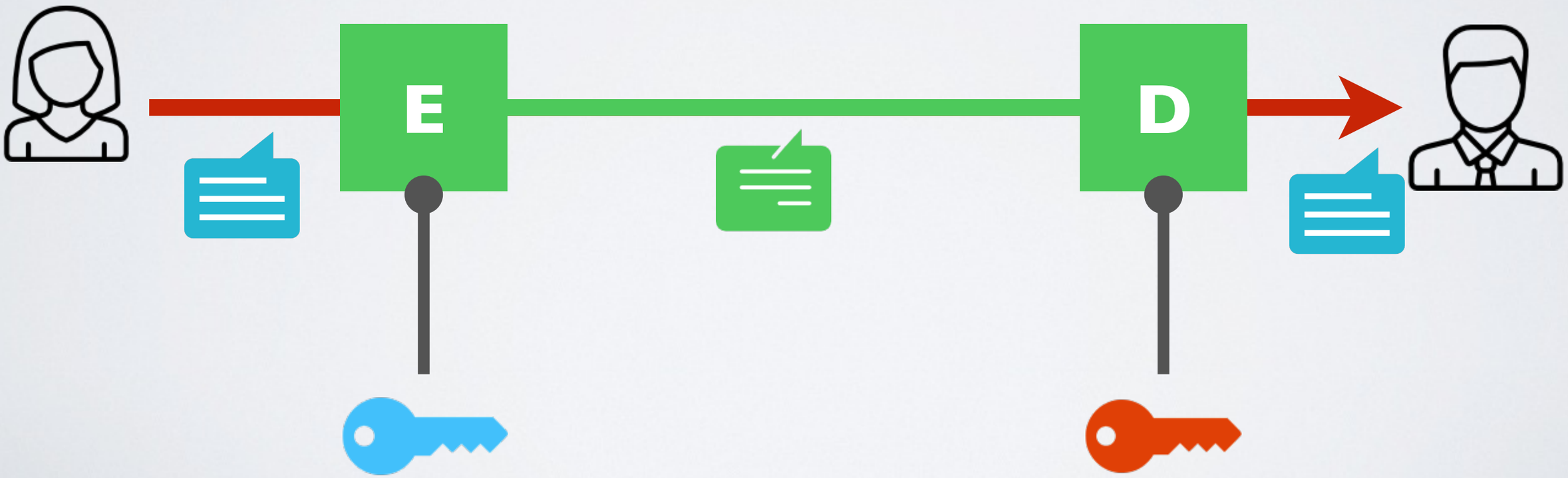
Asymmetric Encryption

Thierry Sans

Asymmetric encryption
a.k.a Public Key Cryptography

- ➡ The public key for encryption
- ➡ The private key for decryption

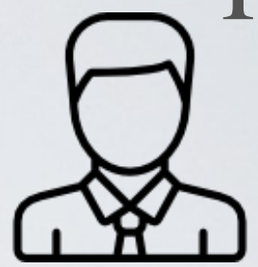


Asymmetric keys

$\mathbf{Ks_A}, \mathbf{Kp_A}$

$\mathbf{Kp_A}$

$\mathbf{Kp_A}$

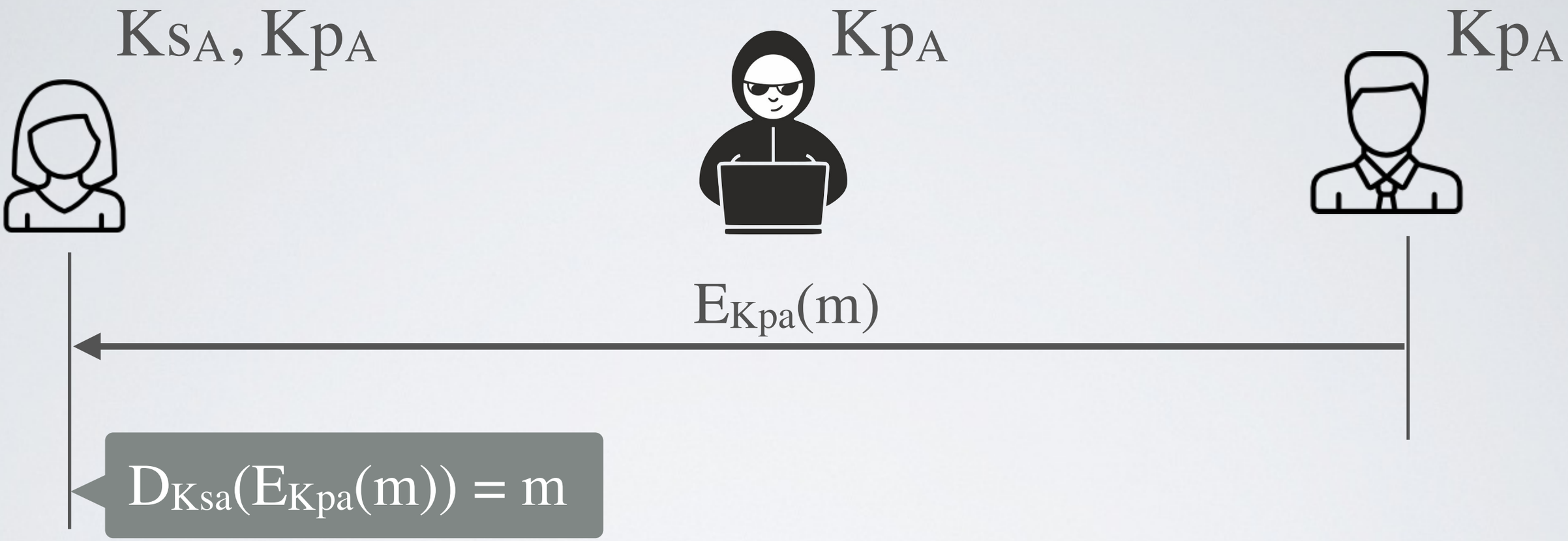


Alice generates a pair of asymmetric keys

- $\mathbf{Ks_A}$ is the secret key that Alice keeps for herself
- $\mathbf{Kp_A}$ is the public key that Alice gives to everyone (even Mallory)

➡ These two keys $\mathbf{Ks_A}$ and $\mathbf{Kp_A}$ work together

Asymmetric encryption for **confidentiality**



Bob encrypts a message $\mathbf{m}$ with Alice's public key $\mathbf{Kp_A}$

➡ Nobody can decrypt $\mathbf{m}$, except Alice with her private key $\mathbf{Ks_A}$

✓ Confidentiality without the need to exchange a secret key

Functional Requirements

$D_{Ks}(E_{Kp}(m)) = m$ and $D_{Kp}(E_{Ks}(m)) = m$ for every pair (Kp, Ks)

- ✓ Generating a pair (Kp, Ks) is easy to compute (polynomial)
- ✓ Encryption is easy to compute (either polynomial or linear)
- ✓ Decryption is easy to compute (either polynomial or linear)
- ◉ Finding a matching key Ks for a given Kp is hard (exponential)
- ◉ Decryption without knowing the corresponding key is hard (exponential)

RSA

RSA - Rivest, Shamir and Alderman

Key Size	1024 - 4096
Speed	~ factor of 10^6 cycles / byte • Key generation: 10 - 100 ms • Encryption: 0.2 - 2 ms • Decryption: 5 - 10 ms
Mathematical Foundation	Prime number theory

Number Theory - Prime numbers

Prime Numbers

- p is prime if 1 and p are its only divisors e.g $3, 5, 7, 11$ …
- p and q are relatively prime (a.k.a. coprime) if $\gcd(p,q) = 1$

  e.g $\gcd(4,5) = 1$

➡ There are infinitely many primes

Euler-Fermat Theorem

If $n = p \,.\, q$ and $z = (p-1).(q-1)$

and a such that a and n are relative primes

Then $a^z \equiv 1 \pmod{n}$

Computational Complexity

Easy problems with prime numbers

- Generating a prime number p
- Addition, multiplication, exponentiation
- Inversion, solving linear equations

Hard problem with prime numbers

- Factoring primes
  e.g. given $\mathbf{n}$ find $\mathbf{p}$ and $\mathbf{q}$ such that $\mathbf{n = p \cdot q}$

RSA - generating the key pair

1. Pick $\mathbf{p}$ and $\mathbf{q}$ two large prime numbers and calculate $\mathbf{n = p \,.\, q}$ (see primality tests)
2. Compute $\mathbf{z = (p\text{-}1).(q\text{-}1)}$
3. Pick a prime number $\mathbf{e < z}$ such that $\mathbf{e}$ and $\mathbf{z}$ are relative primes

➡ (e,n) is the **public key**

4. Solve the linear equation $\mathbf{e * d = 1\ (\ mod\ z\ )}$ to find $\mathbf{d}$

➡ (d,n) is the **private key**
however p and q must be kept secret too

RSA - encryption and decryption

Given $\mathbf{Kp} = (\mathbf{e}, \mathbf{n})$ and $\mathbf{Ks} = (\mathbf{d}, \mathbf{n})$

➡ Encryption : $E_{kp}(m) = m^e \bmod n = c$

➡ Decryption : $D_{ks}(c) = c^d \bmod n = m$

➡ **$(m^e)^d \bmod n = (m^d)^e \bmod n = m$**

The security of RSA

RSA Labs Challenge : factoring primes set

Key length	Year	Time
140	1999	1 month
155	1999	4 months
160	2003	20 days
200	2005	18 months
768	2009	3 years

Challenges are no longer active

ECC

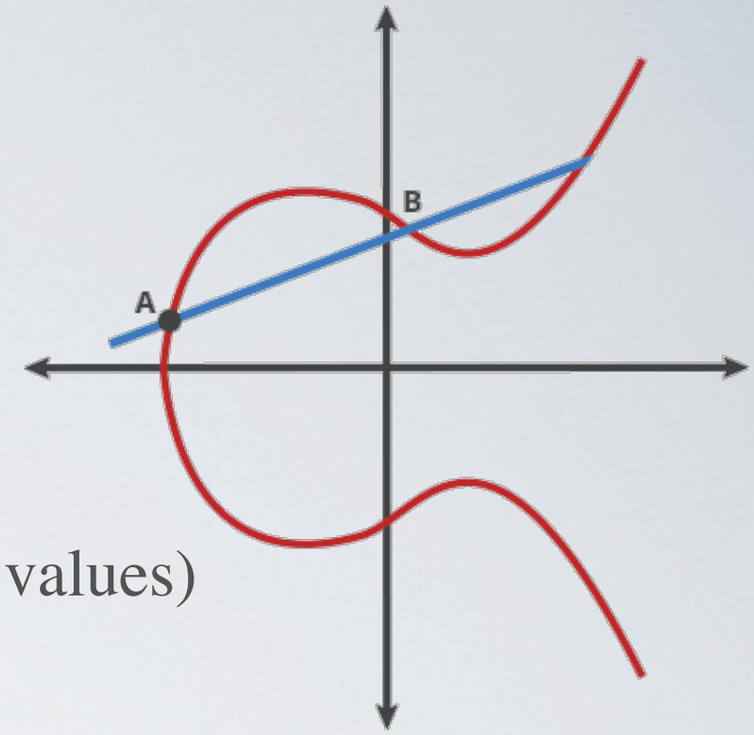
ECC - Elliptic-Curve Cryptography

Key Size	256 or 448 bits
Speed	~ factor of 10^6 cycles / operation • Key generation: 1 - 5 ms (way faster than RSA) • Encryption: 1 - 5 ms • Decryption: 1 - 5 ms
Mathematical Foundation	Elliptic curves over finite fields

Main ECC Standards

	secp256k1	curve25519	curve448
Year	2000	2005	2014
Inventor	Standards for Efficient Cryptography Group (SECG)	Daniel J. Bernstein	Mike Hamburg
Key Size	256	256	448
Applications	Bitcoin Ethereum	TLS, TOR Signal Protocol Monero, Zcash	TLS
Performances	+	++	+++

Elliptic Curve Cryptography



Use Elliptic-curve for generating a cryptographic public-key pair
The algorithm is based on two public pieces:

- The curve equation $y^2 = x^3 + ax + b$ (a and b are fixed values)
- The generator point (fixed value)

When generating a key pair

1. the user "choose a random number" (within a given range) as private key
2. then derived the public key from the curve

✓ Smaller key sizes: 256 bits EC keys has the same entropy as RSA 3072 bits

✓ Can be used for digital signature (ECDSA algorithm)

✓ Can be used for key agreement (ECDH algorithm)

https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/

Symmetric vs Asymmetric

Key length and Key n-bit security

- RSA has very long keys, 1024, 2048 and 4096 are common
- ECC has shorter keys, 256 and 448 are common
- Is it more secure than symmetric crypto with key lengths of 56, 128, 192, 256 ?

➡ Key lengths **do not compare !**

RSA	ECC	Effective key length
1,024		80
2,048		112
3,072	256	128
4096		140
15,360	448	224 ~ 256

Asymmetric vs Symmetric

	Symmetric	Asymmetric
pro	Fast	No key agreement
cons	Key agreement	Very slow

The best of both worlds

- Use asymmetric encryption to encrypt a shared key (or hash)
- Use symmetric encryption to encrypt message

$$E_{Kp}(m) = RSA_{Kp}(k), AES_k(m)$$

Naive approach

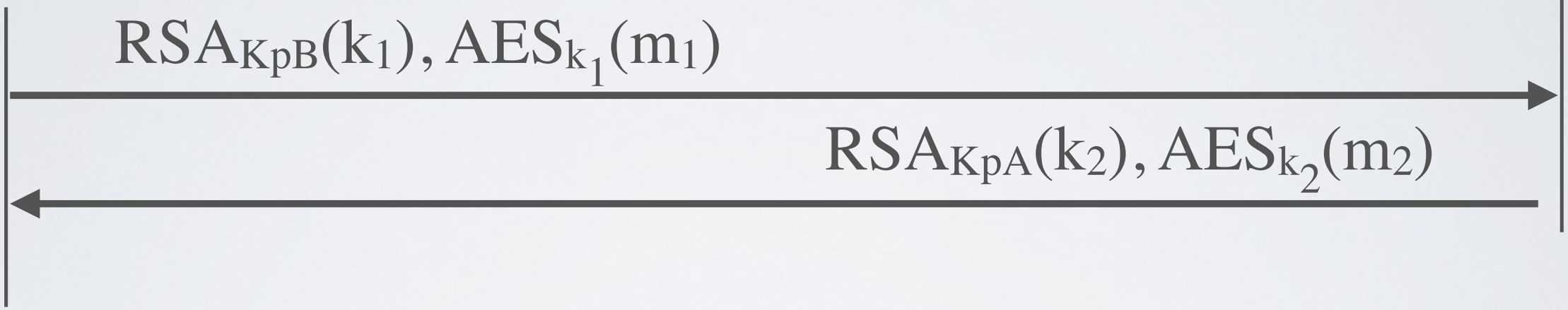
But not perfect yet

Ks_A, Kp_A

Kp_A, Kp_B

Ks_B, Kp_B

Kp_A



- ✓ Does ensure the confidentiality of the communication
- ◉ Does not authenticate Alice or Bob

Asymmetric encryption for key exchange

Should we use asymmetric encryption for key exchange?

- ✓ Simple solution for non-interactive protocol (e.g GPG)
- ◉ But not a good solution for interactive protocols