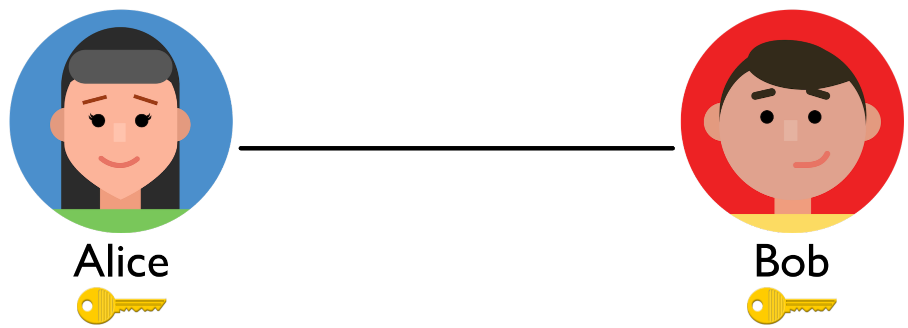
Communication with Shared Secrets

We have several ways for Alice and Bob to send confidential messages, and all require a  as a **shared secret**



How do Alice and Bob get a shared secret in the first place?

It turns out that it's possible to turn private secrets into a shared secret through a *public* conversation!

Public Key Cryptography

Two widely used algorithms to create shared secrets:

- **Diffie-Hellman**
- **RSA**

Both from the 1970s with similar capabilities—but different immediate uses, and RSA dominates for historical and commercial reasons

Public Key Cryptography

Two widely used algorithms to create shared secrets:

- **Diffie-Hellman-Merkel**
- **RSA**

Both from the 1970s with similar capabilities—but different immediate uses, and RSA dominates for historical and commercial reasons

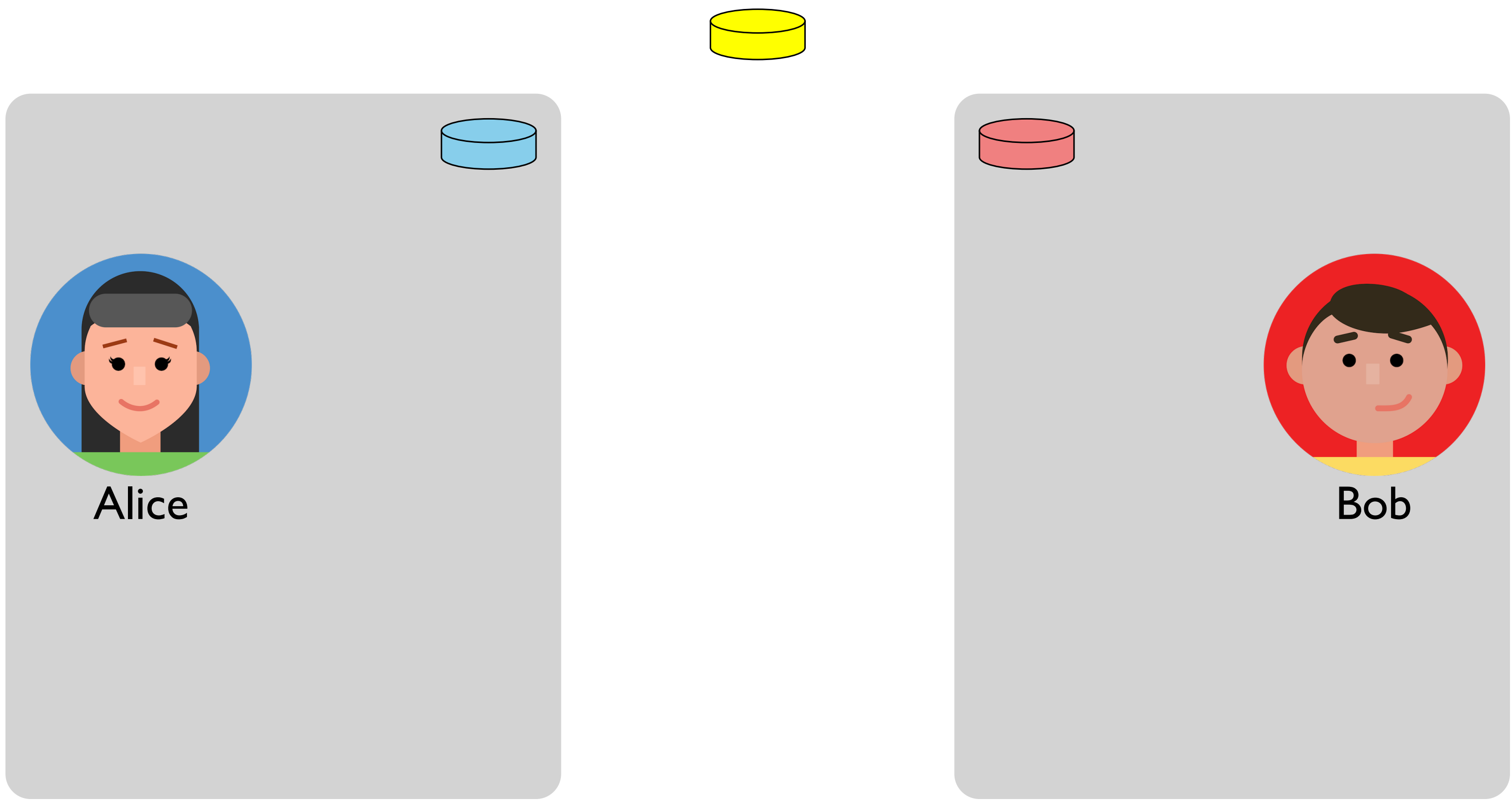
Public Key Cryptography

Two widely used algorithms to create shared secrets:

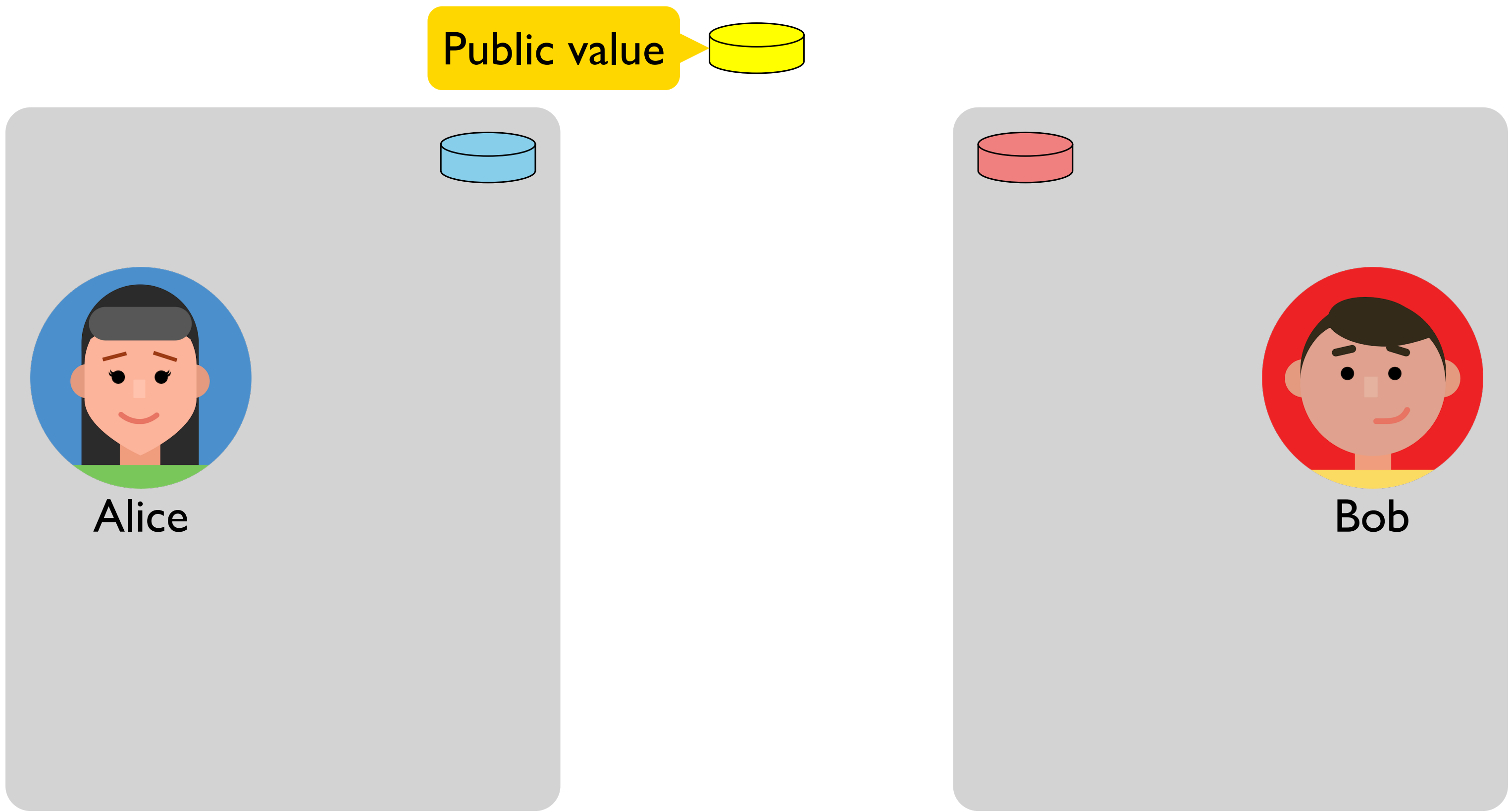
- **Diffie-Hellman-Merkel**
- **Rivest-Shamir-Adelman**

Both from the 1970s with similar capabilities—but different immediate uses, and RSA dominates for historical and commercial reasons

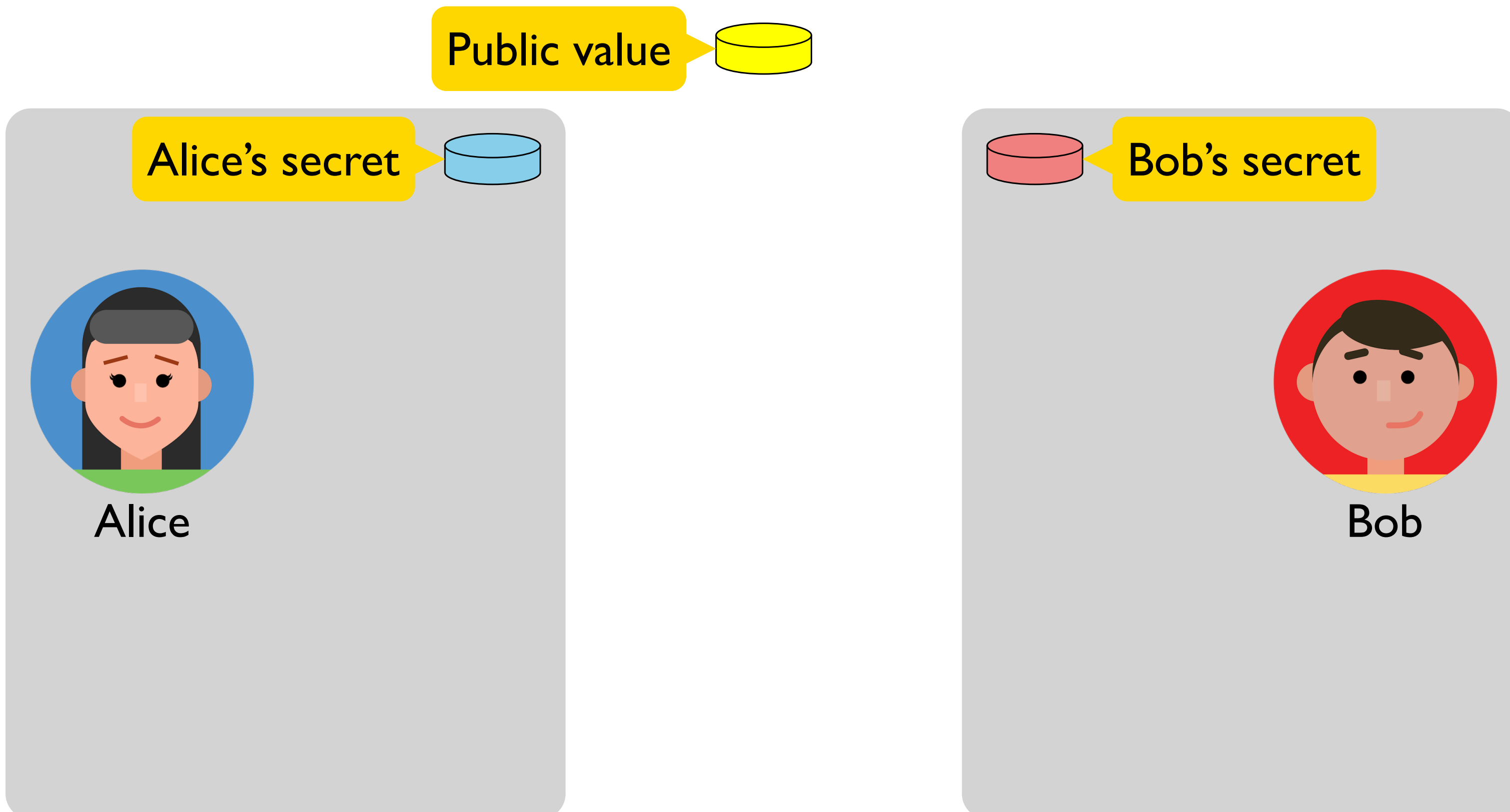
Diffie-Hellman Key Exchange



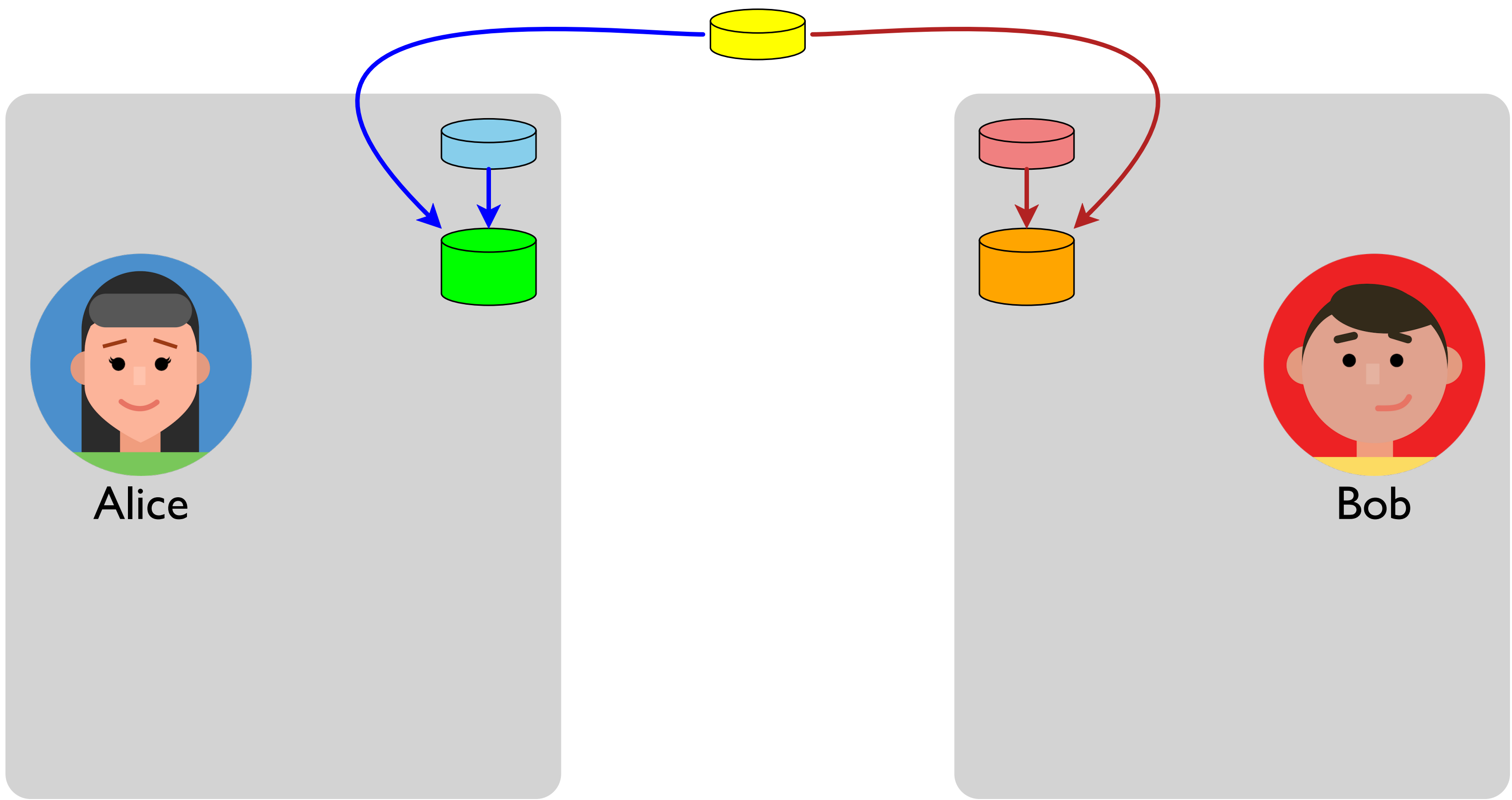
Diffie-Hellman Key Exchange



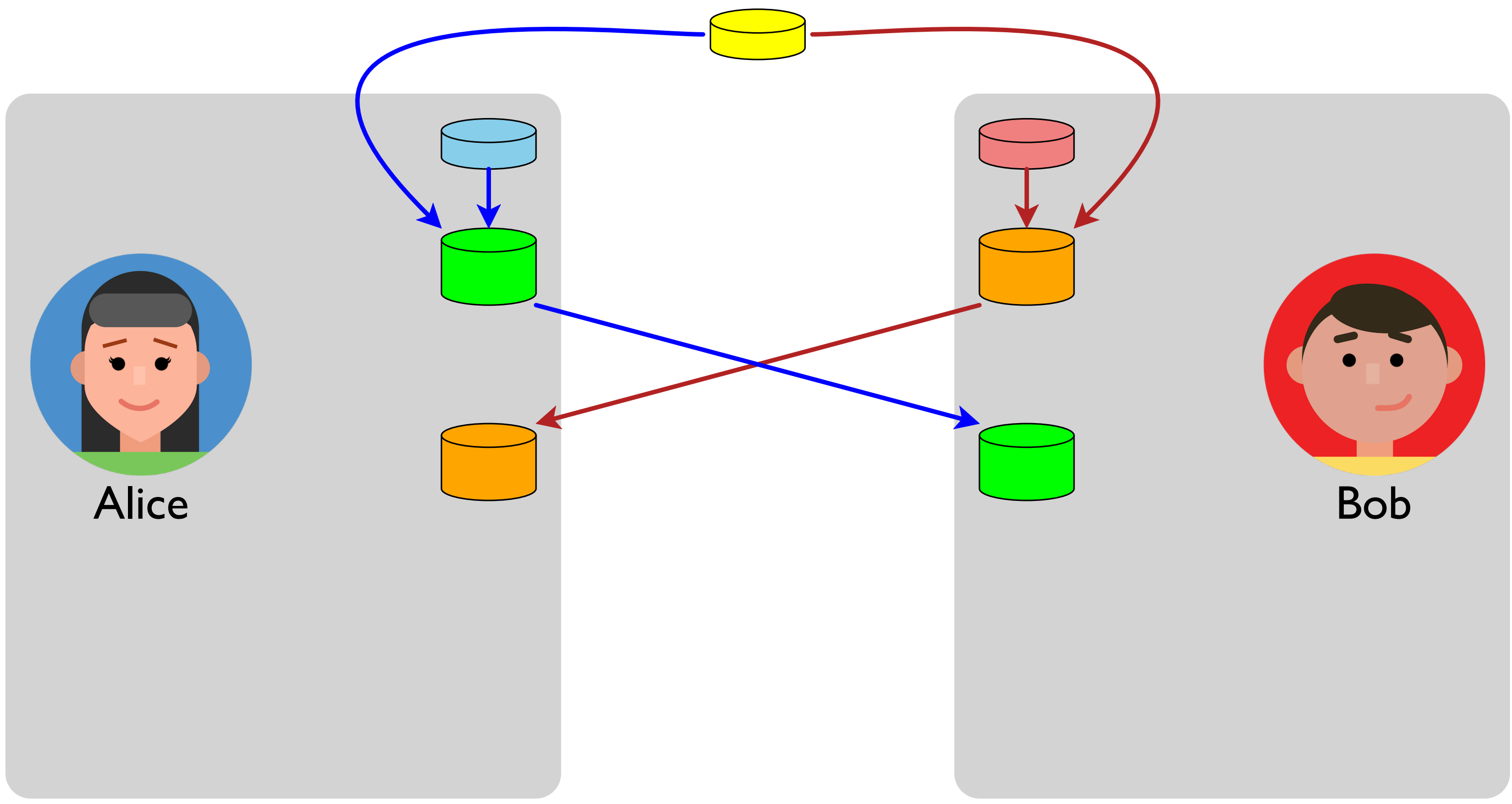
Diffie-Hellman Key Exchange



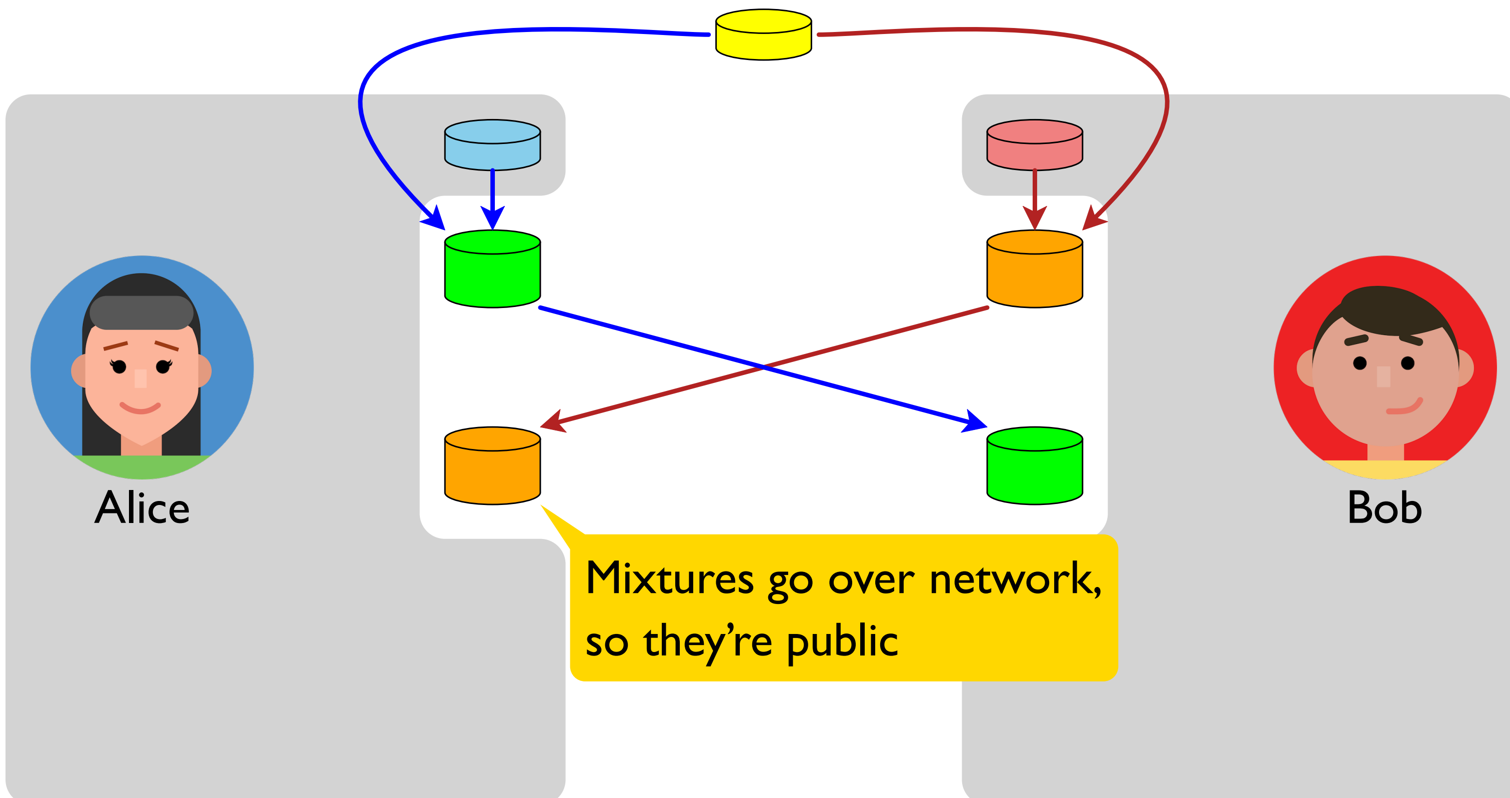
Diffie-Hellman Key Exchange



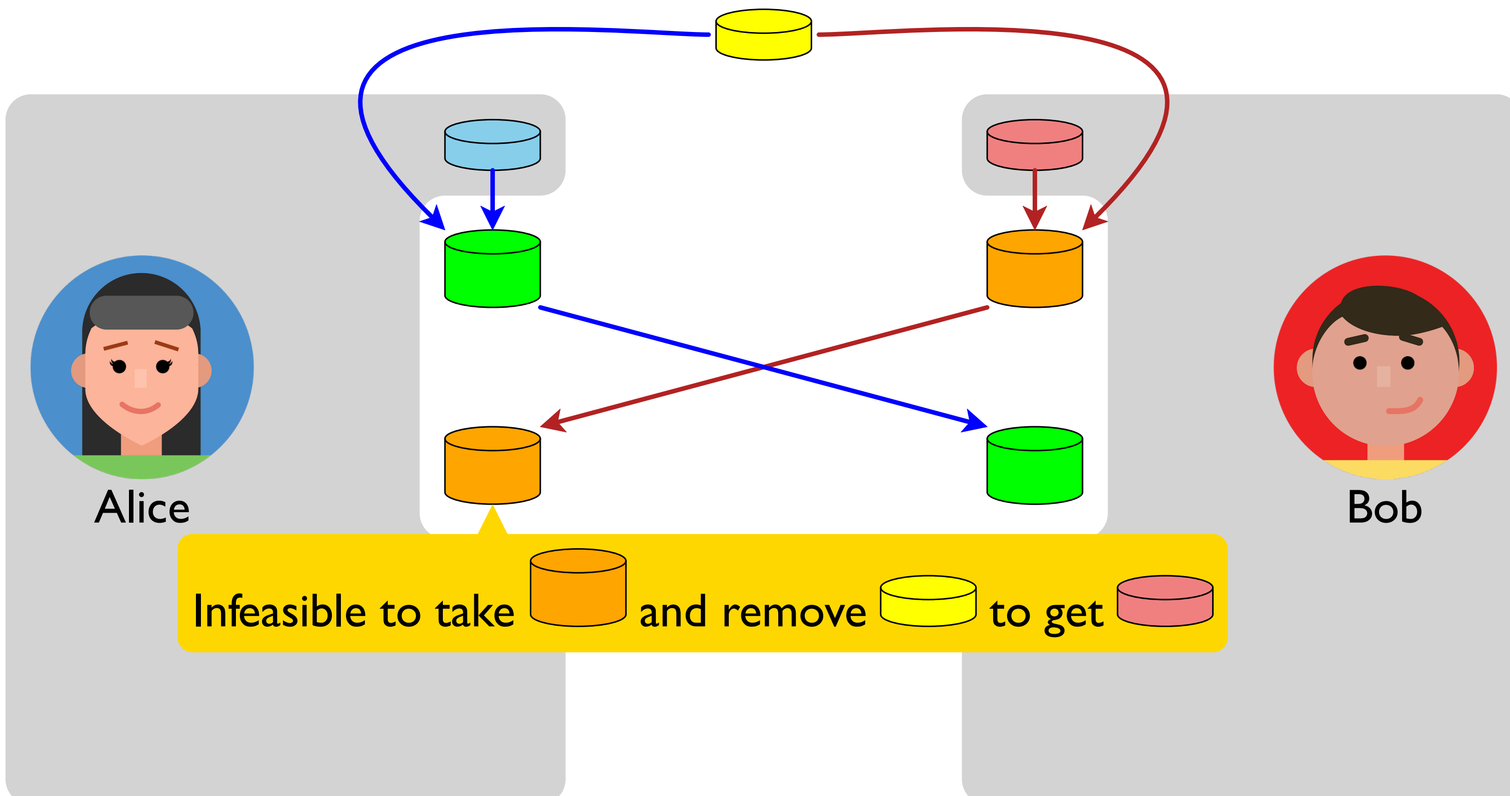
Diffie-Hellman Key Exchange



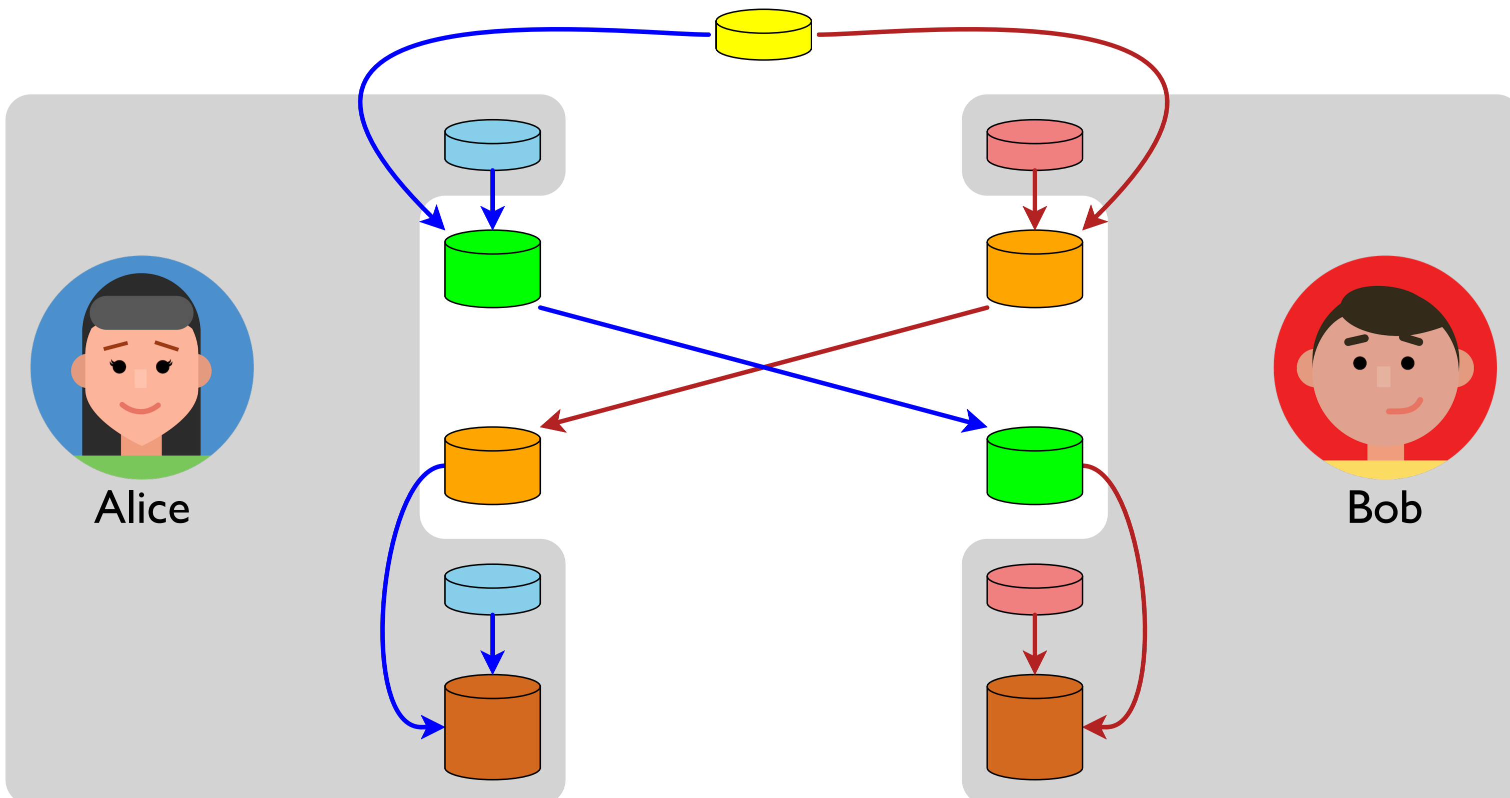
Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange

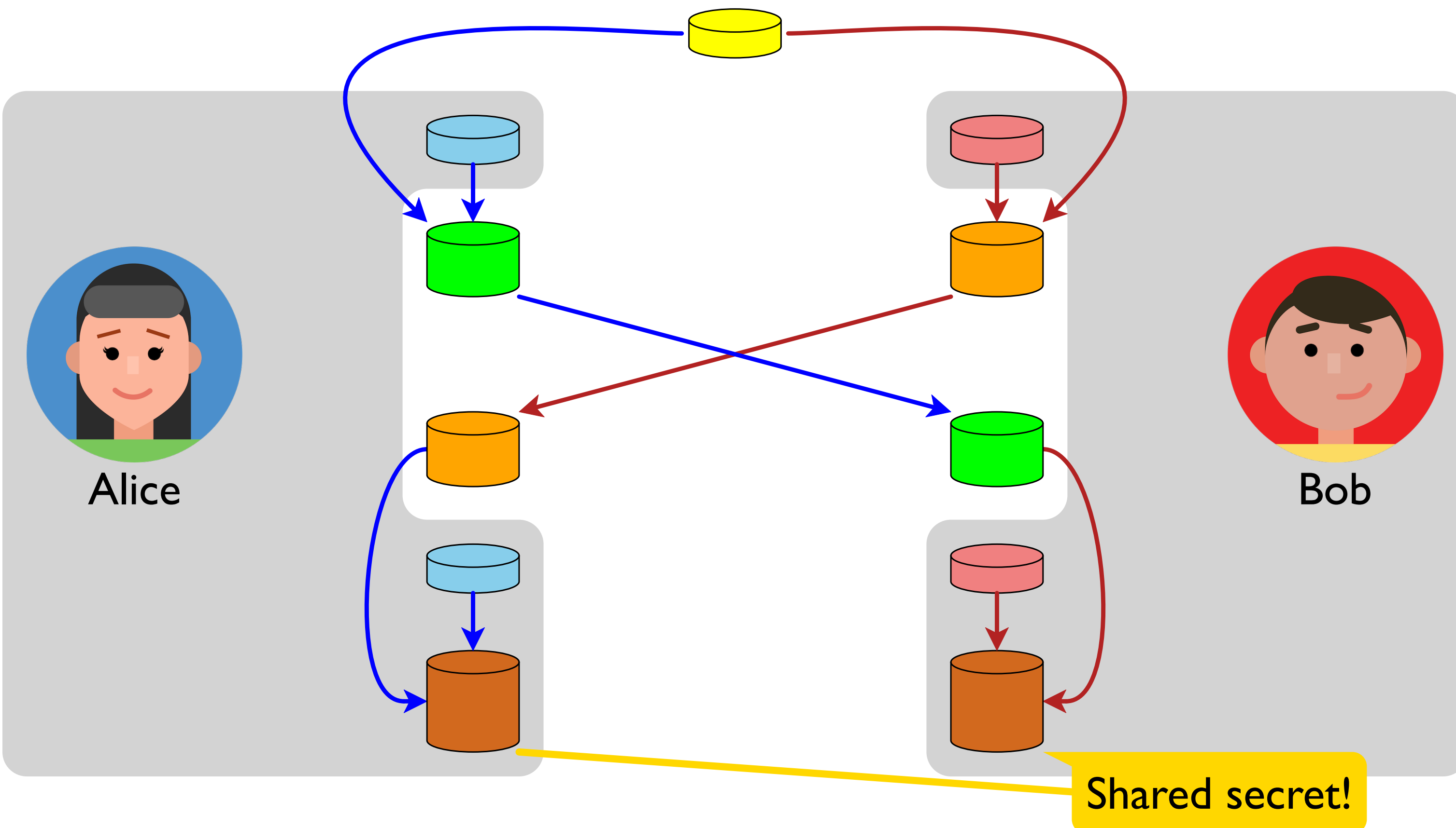


Diffie-Hellman Key Exchange



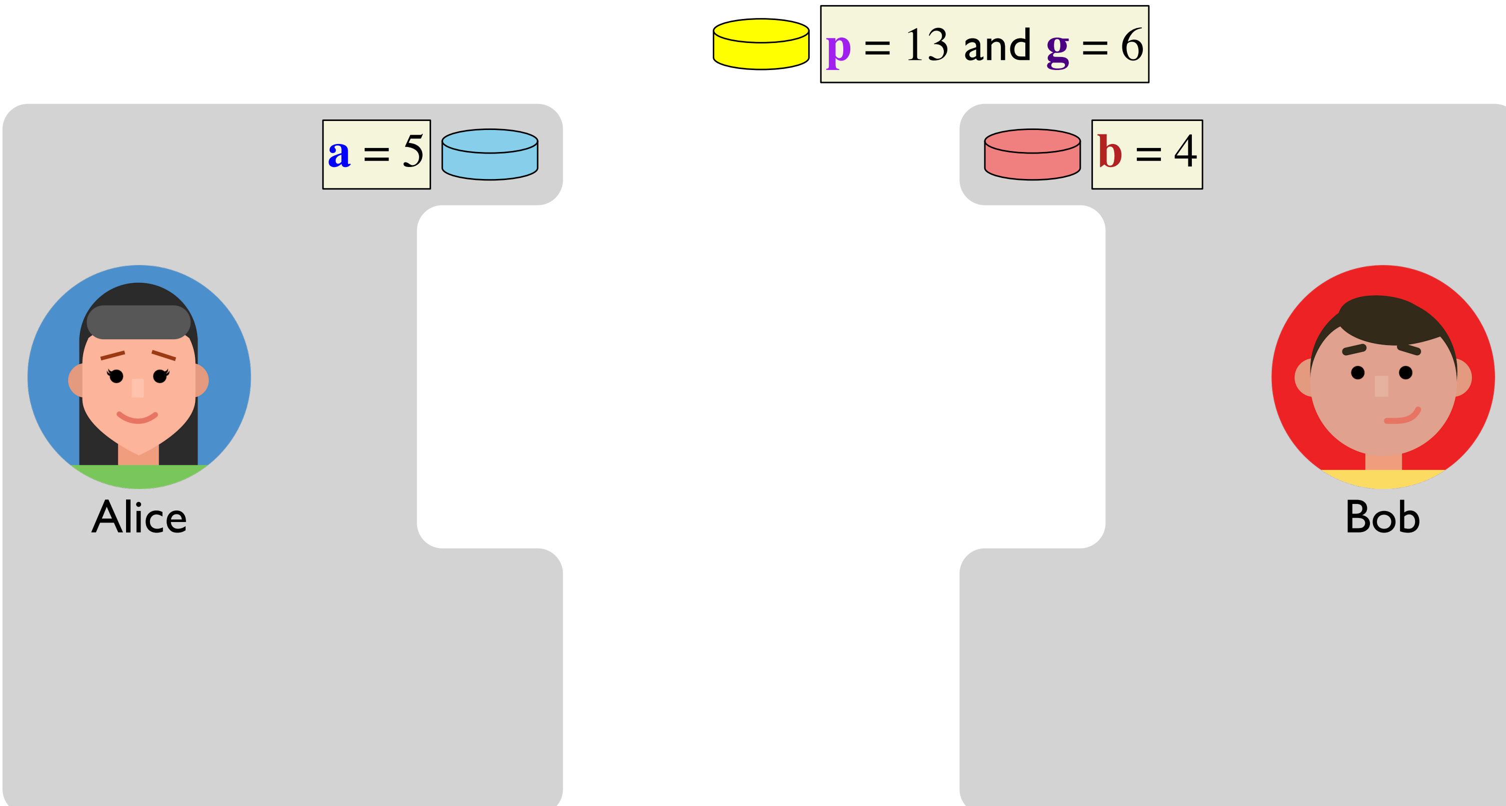
https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

Diffie-Hellman Key Exchange

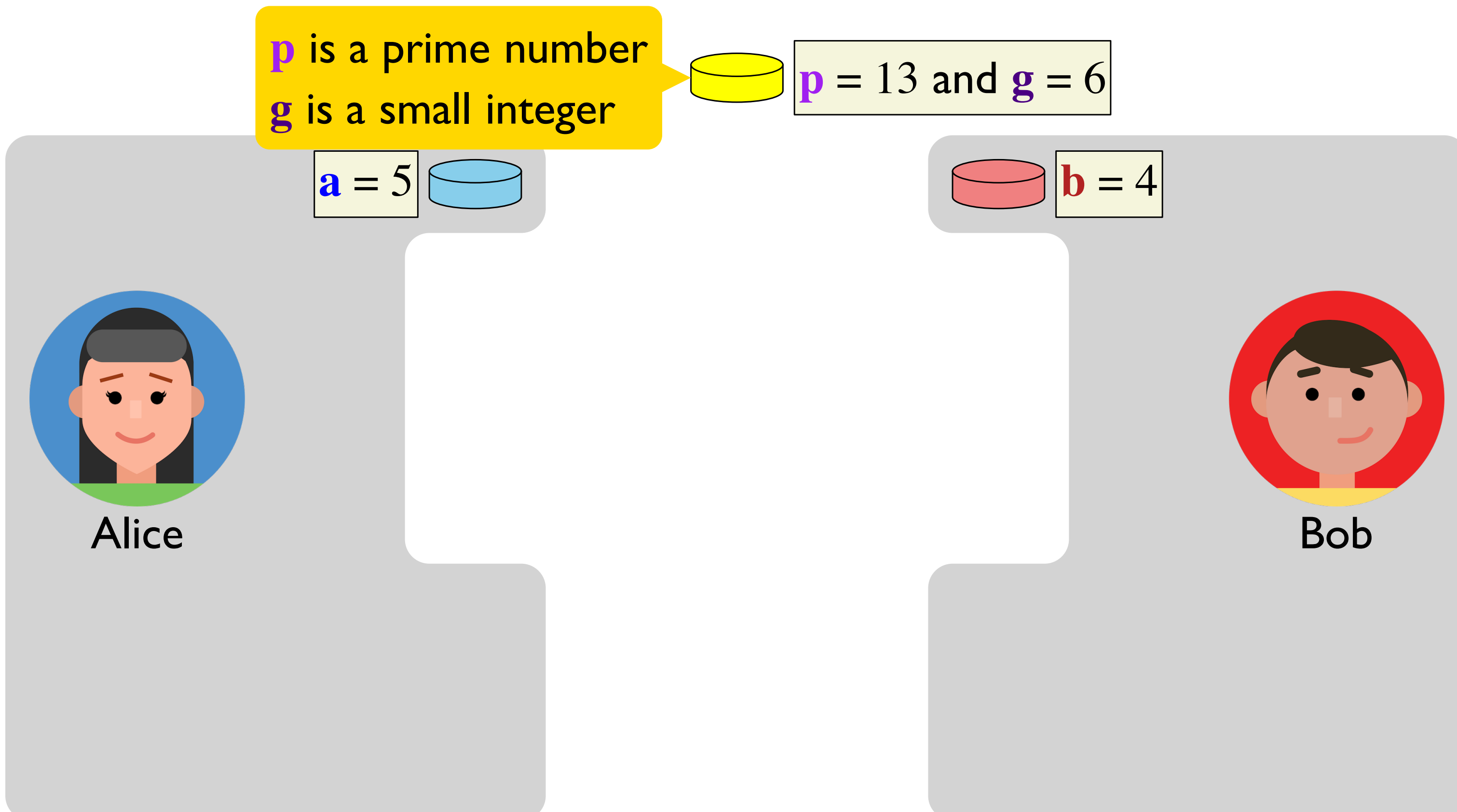


https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange

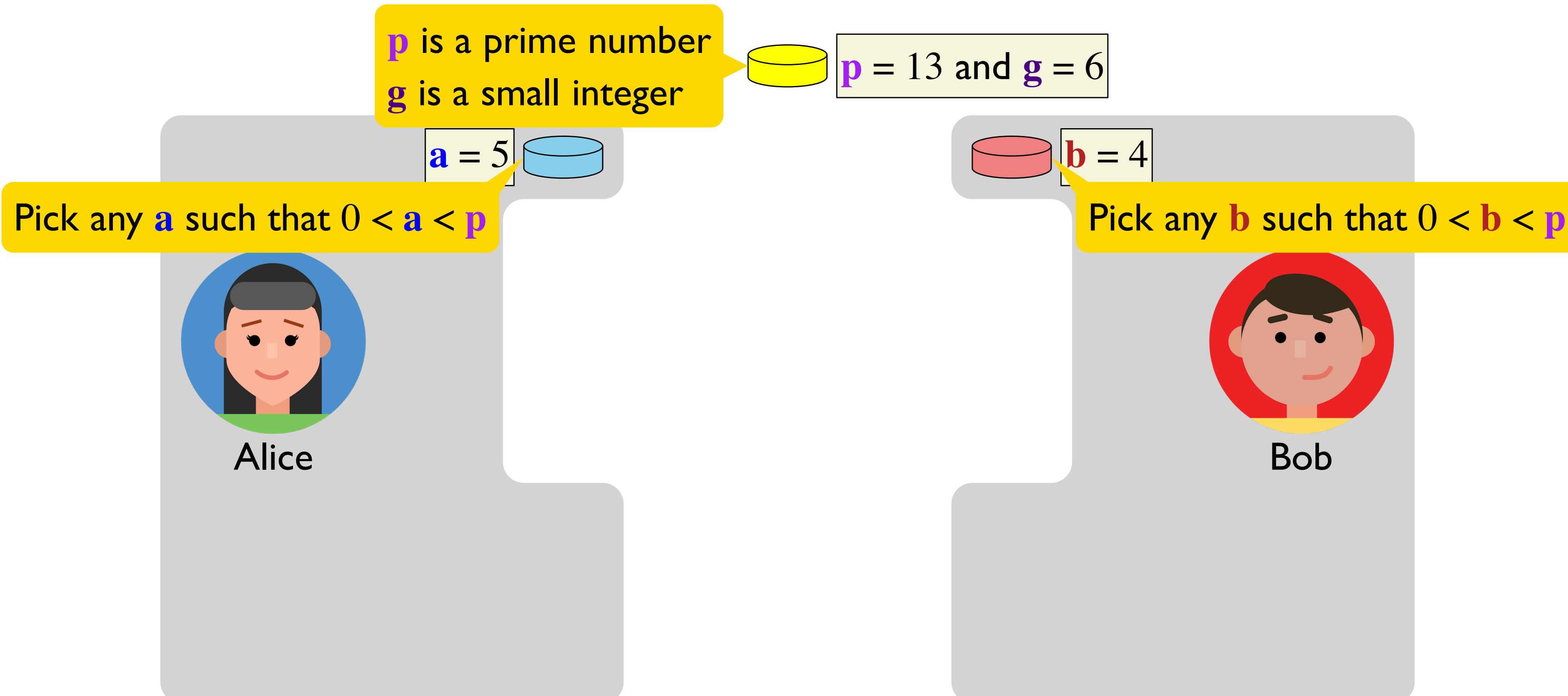
Diffie-Hellman Key Exchange



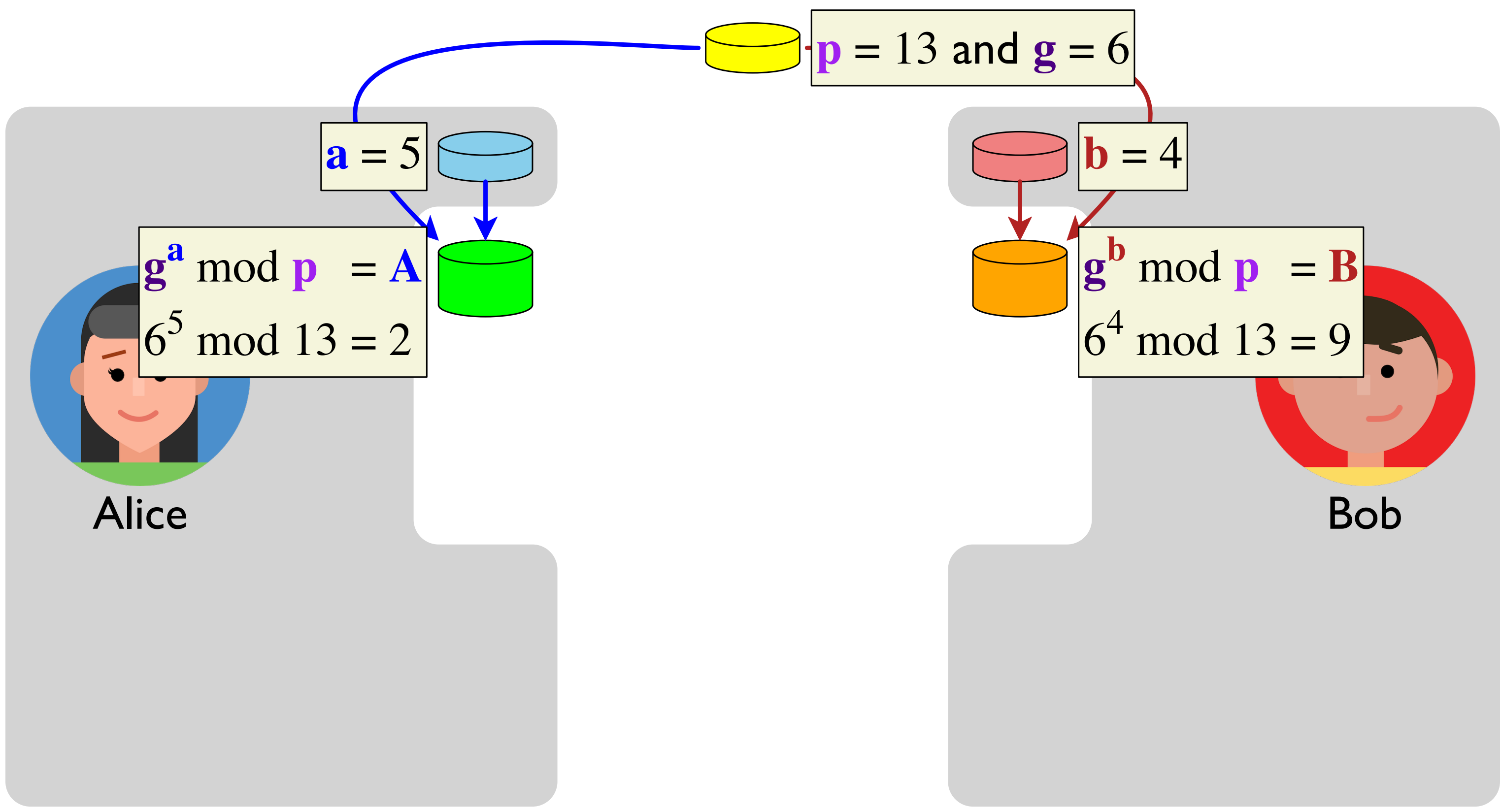
Diffie-Hellman Key Exchange



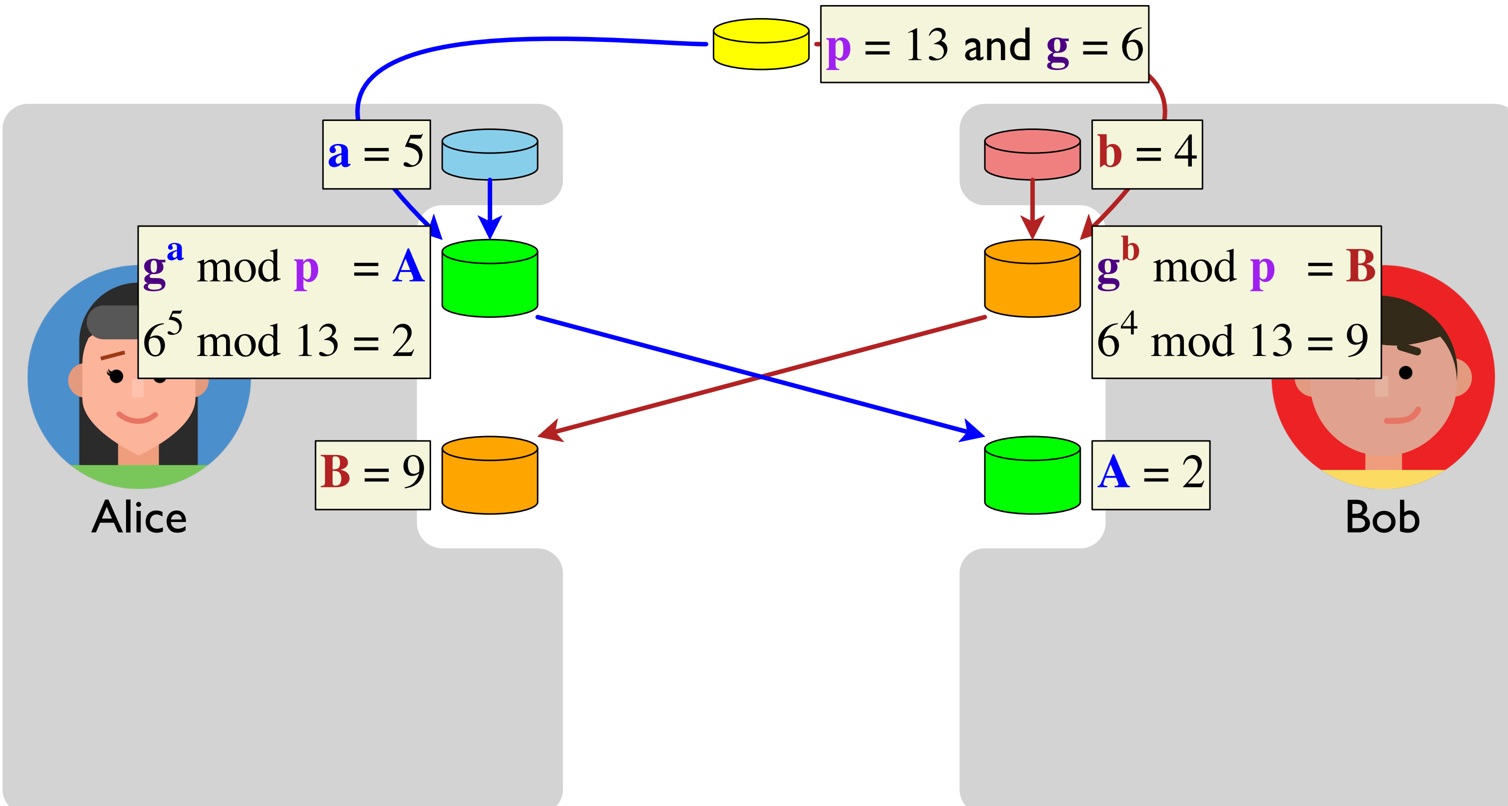
Diffie-Hellman Key Exchange



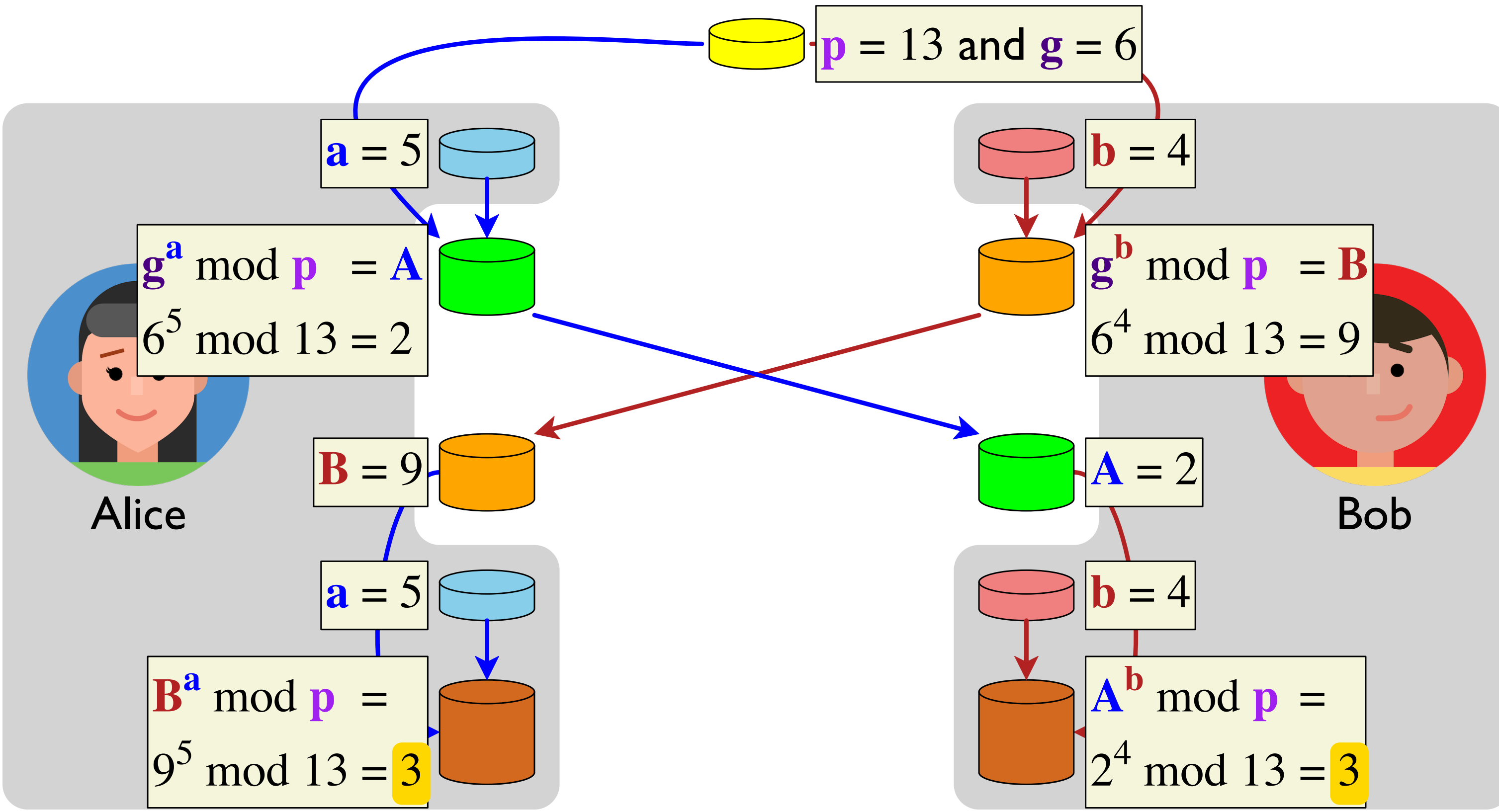
Diffie-Hellman Key Exchange



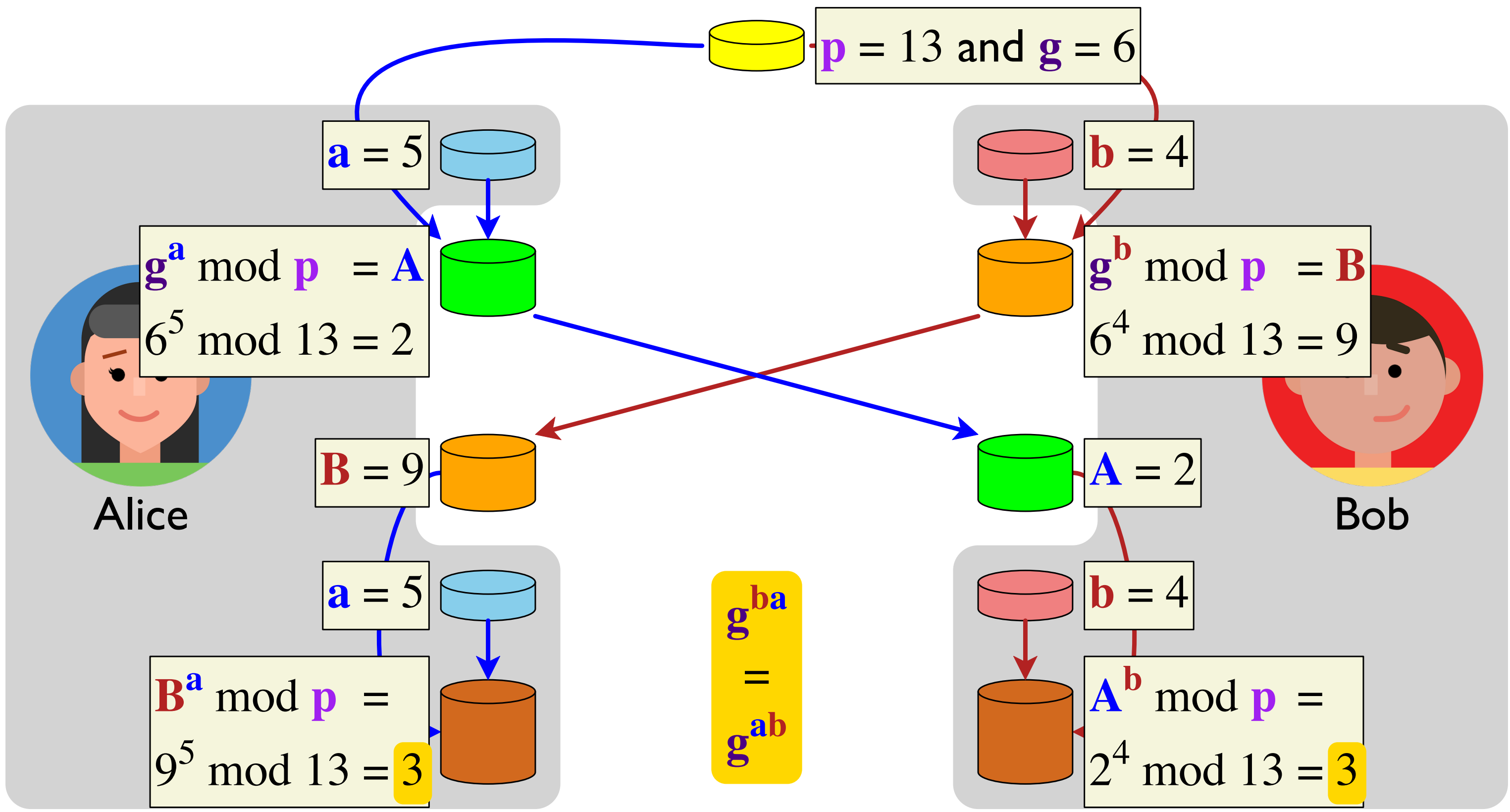
Diffie-Hellman Key Exchange



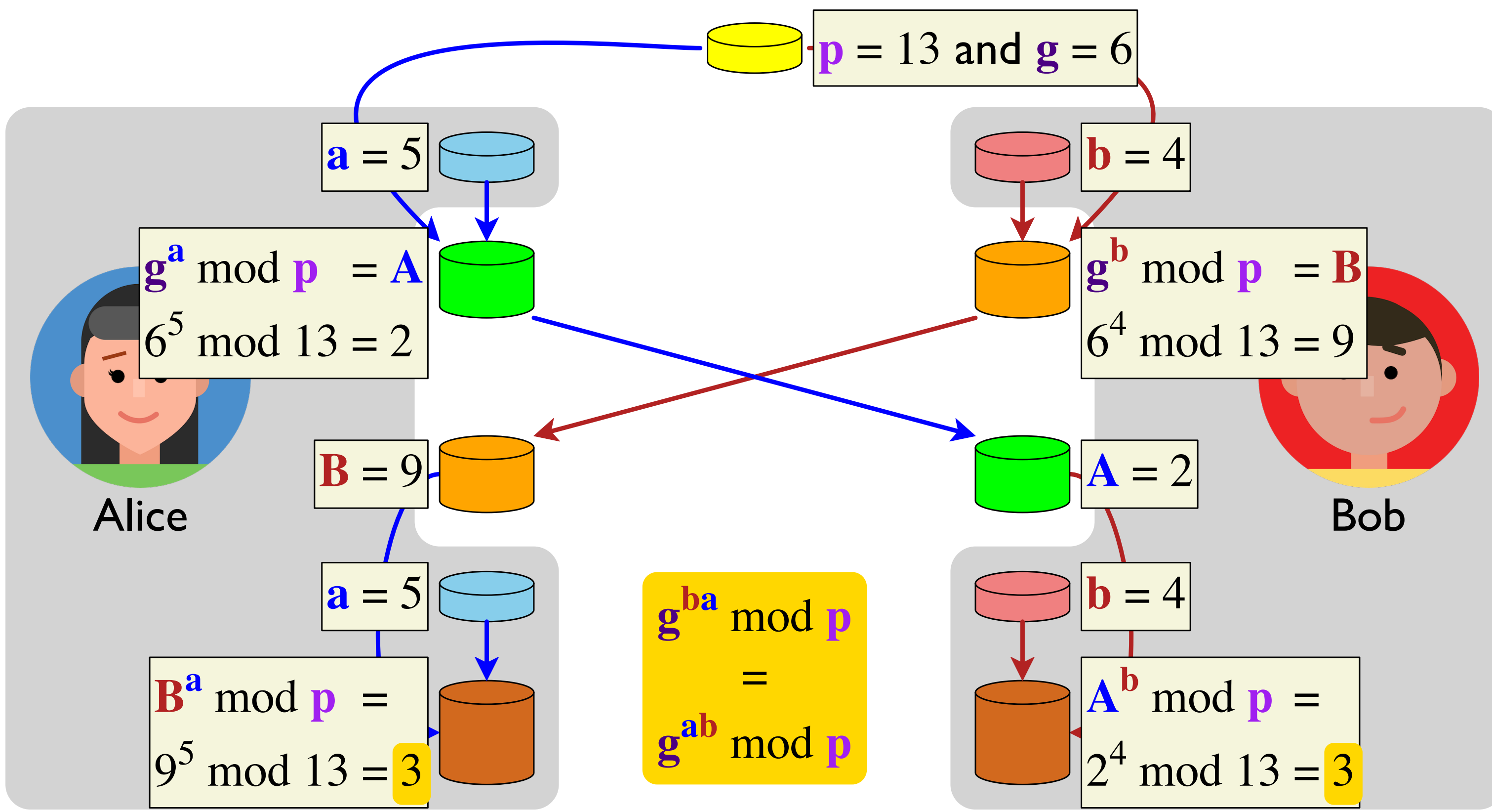
Diffie-Hellman Key Exchange



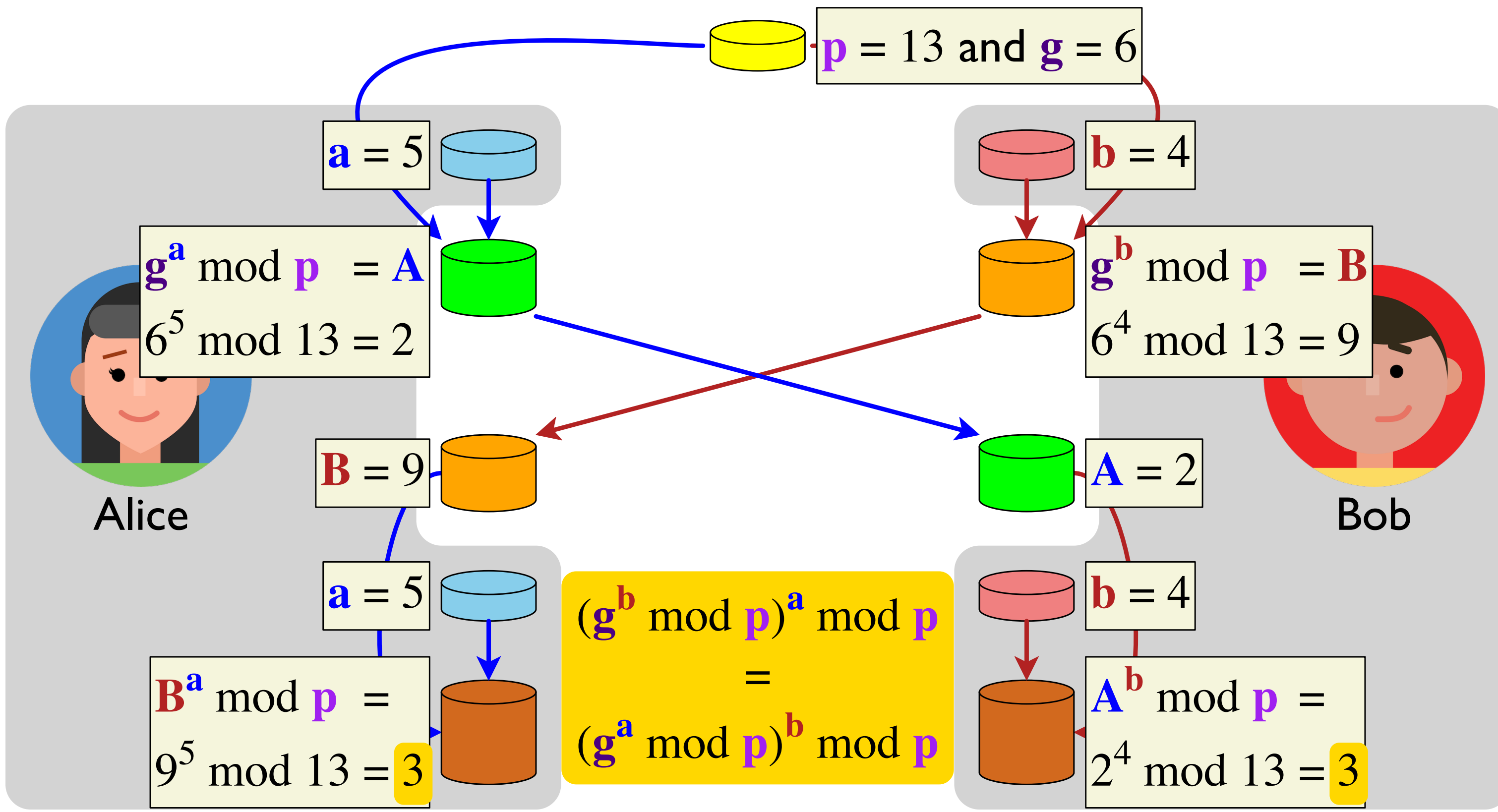
Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange



Discrete Logarithm Problem

For a large **p**, **a**, and **b**, it's infeasible to get from

$$\mathbf{A} = \mathbf{g}^{\mathbf{a}} \bmod \mathbf{p}$$

$$\mathbf{B} = \mathbf{g}^{\mathbf{b}} \bmod \mathbf{p}$$

back to **a** or **b**

"Large" in practice means 1024 to 8192 bits for **p**, **a**, and **b**

At that scale, $\mathbf{g}^{\mathbf{a}}$, $\mathbf{g}^{\mathbf{b}}$, and $\mathbf{g}^{\mathbf{ab}}$ do not remotely fit in in the universe, but the values $\bmod$ **p** are small and can be computed quickly

Discrete Logarithm Problem

For a large $\mathbf{p}$, $\mathbf{a}$, and $\mathbf{b}$, it's infeasible to get from

$$\mathbf{A} = \mathbf{g}^{\mathbf{a}} \bmod \mathbf{p}$$

$$\mathbf{B} = \mathbf{g}^{\mathbf{b}} \bmod \mathbf{p}$$

back to $\mathbf{a}$ or $\mathbf{b}$

"Large" in practice means 1024 to 8192 bits for $\mathbf{p}$, $\mathbf{a}$, and $\mathbf{b}$

At that scale, $\mathbf{g}^{\mathbf{a}}$, $\mathbf{g}^{\mathbf{b}}$, and $\mathbf{g}^{\mathbf{ab}}$ do not remotely fit in in the universe, but the values $\mathrm{mod}\ \mathbf{p}$ are small and can be computed quickly

$x^2 \bmod \mathbf{p} = (x \bmod \mathbf{p})^2 \bmod \mathbf{p}$

⇒ divide and conquer

Internet Key Exchange (IKE)

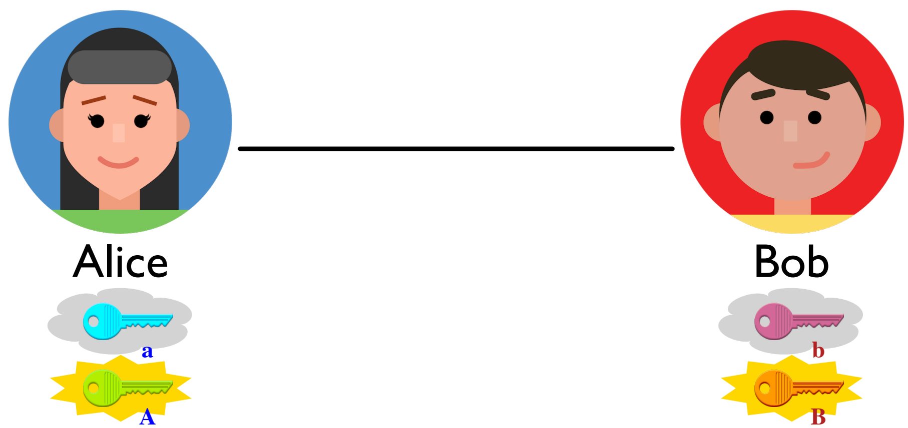
RFC 3526's 2048-bit $\mathbf{p}$ with $\mathbf{g} = 2$:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AACAA68 FFFFFFFF FFFFFFFF
```

Public Key Cryptography

General idea is that keys come in pairs:

private and public



Public Key Cryptography

General idea is that keys come in pairs:

private and public

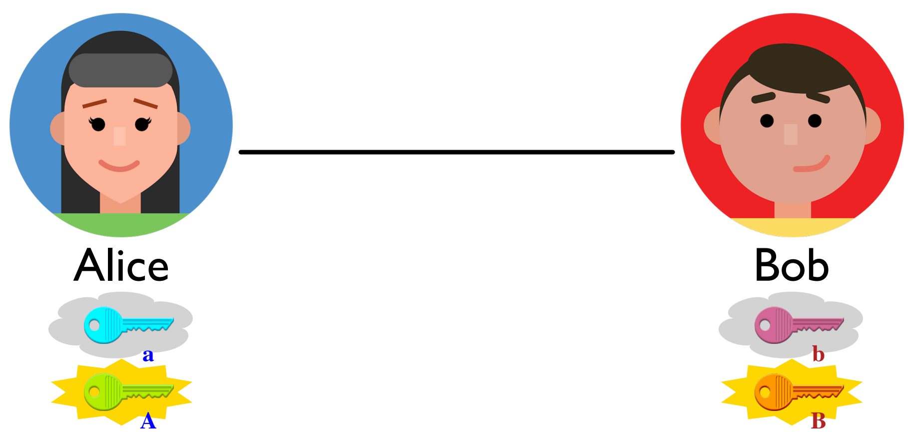
decrypt

Public Key Cryptography

General idea is that keys come in pairs:

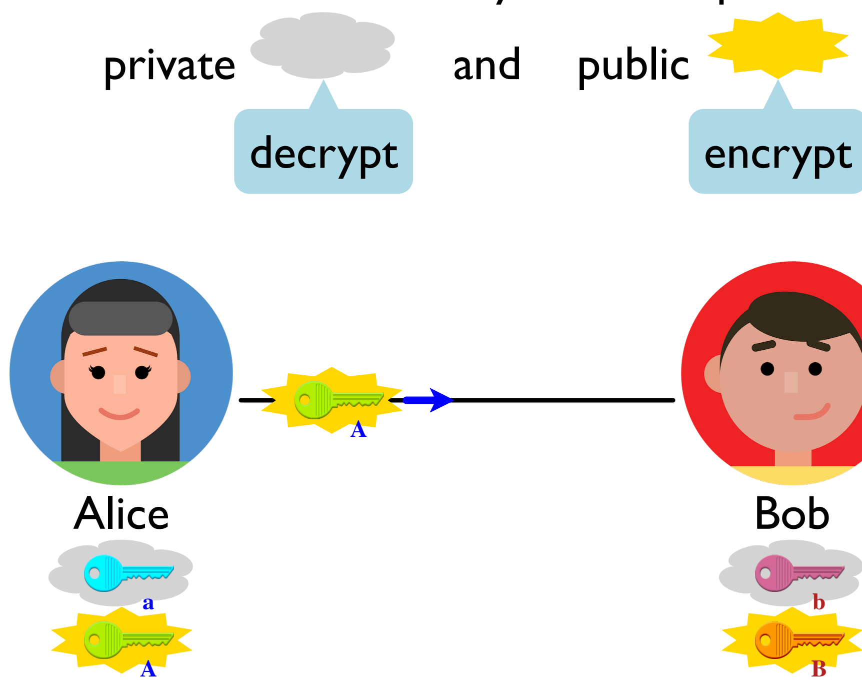
private and public

decrypt encrypt



Public Key Cryptography

General idea is that keys come in pairs:



Public Key Cryptography

General idea is that keys come in pairs:

private and public

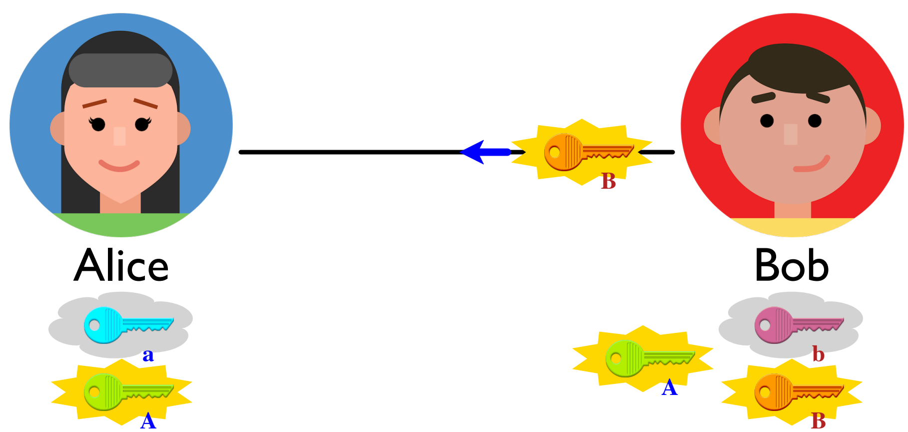
decrypt encrypt

Public Key Cryptography

General idea is that keys come in pairs:

private and public

decrypt encrypt



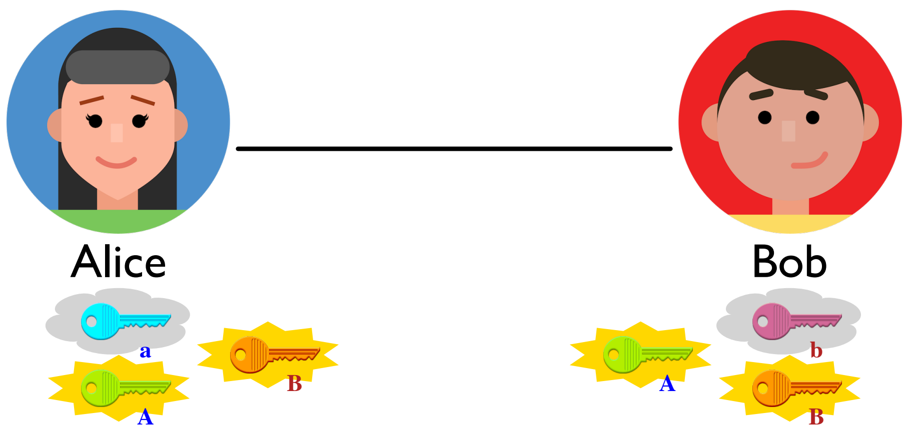
Public Key Cryptography

General idea is that keys come in pairs:

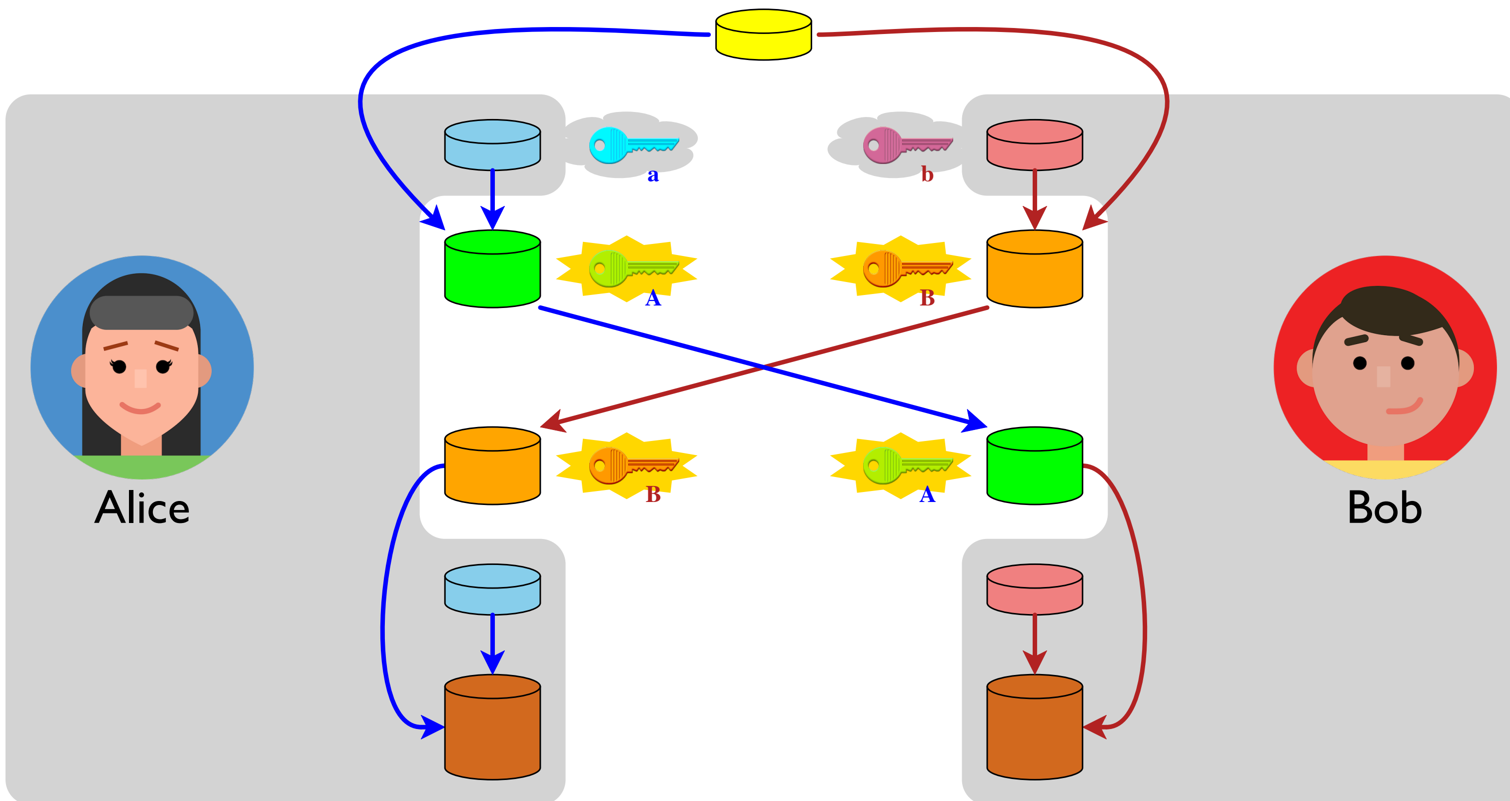
private and public

decrypt

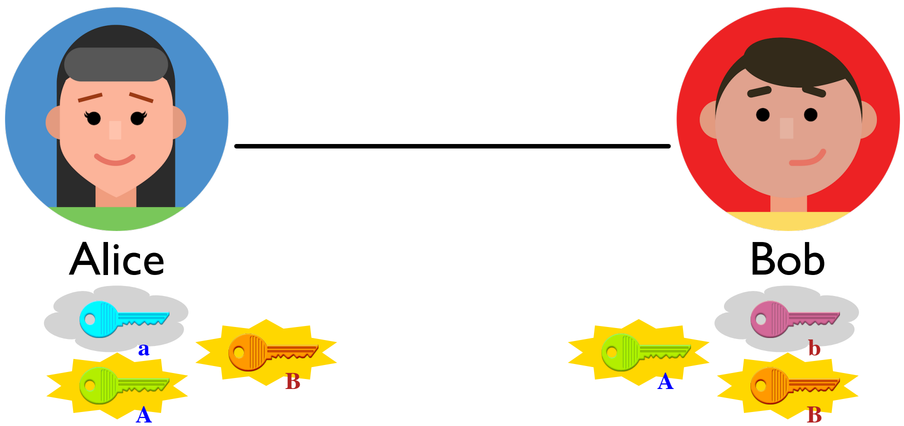
encrypt



Diffie-Hellman Key Exchange as Public Key Infrastructure

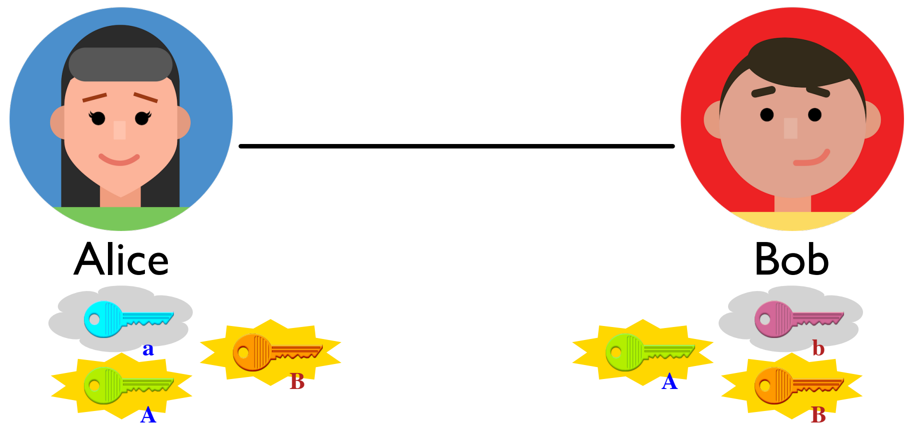


Public Key Cryptography



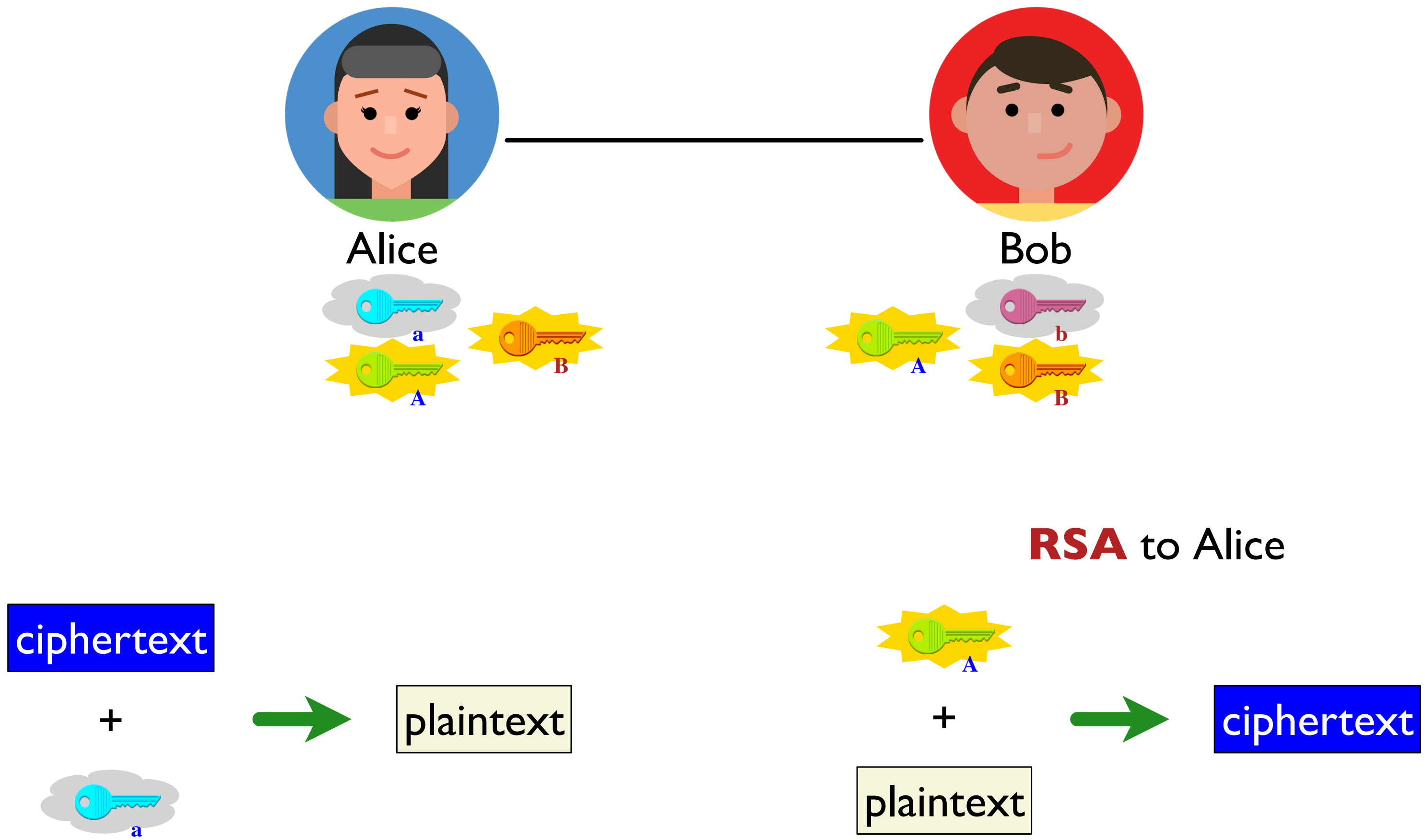
Diffie-Hellman

Public Key Cryptography

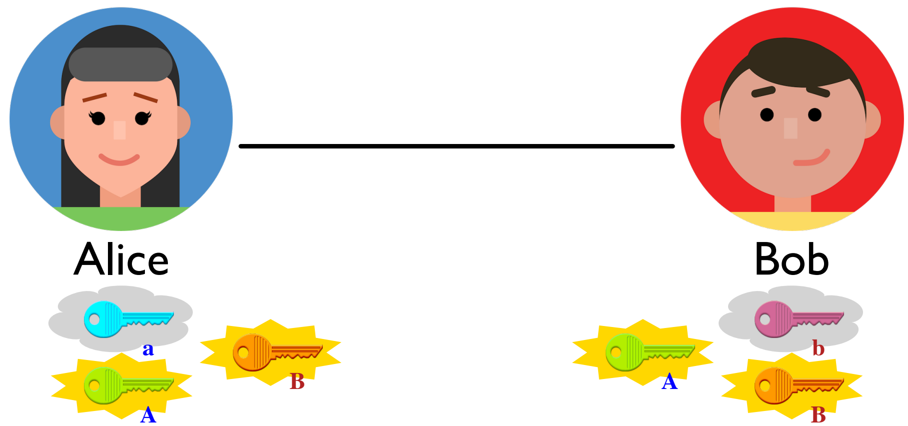


RSA to Bob

Public Key Cryptography



Public Key Cryptography

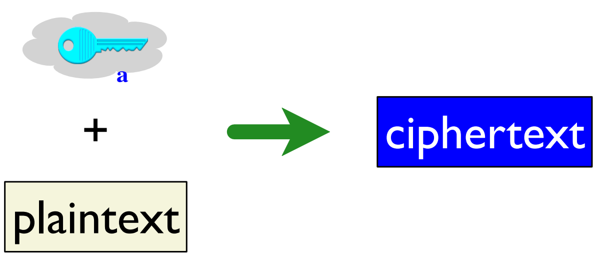


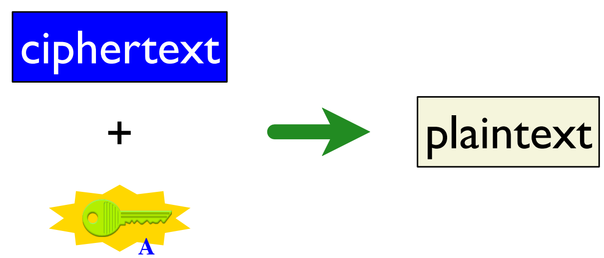
RSA *signing* by Alice

Public Key Cryptography

RSA public and private keys each work in both directions, so they can be used for both **confidentiality** and **authentication**

RSA *signing* by Alice





RSA

Alice picks

- **p** and **q** as large, random, k-bit prime numbers
- **e** as relatively prime to (**p**-1) × (**q**-1)

RSA

Alice picks

Something like 1024 to 8192

- **p** and **q** as large, random, k-bit prime numbers
- **e** as relatively prime to $(\mathbf{p}-1) \times (\mathbf{q}-1)$

RSA

Easy to generate with high probability due to density of prime numbers and a quick “probably prime” test

Alice picks

- **p** and **q** as large, random, k-bit prime numbers
- **e** as relatively prime to $(p\text{-}1) \times (q\text{-}1)$

RSA

Alice picks

- **p** and **q** as large, random, k-bit prime numbers
- **e** as relatively prime to (**p**-1) × (**q**-1)

Even easier: arbitrary number plus a check that GCD is 1

RSA

Alice picks

- **p** and **q** as large, random, k-bit prime numbers
- **e** as relatively prime to (**p**-1) × (**q**-1)

Find **d** so that (**e** × **d**) mod ((**p**-1) × (**q**-1)) = 1

Define **N** = **p** × **q**

RSA

Alice picks

- $\mathbf{p}$ and $\mathbf{q}$ as large, random, k-bit prime numbers
- $\mathbf{e}$ as relatively prime to $(\mathbf{p}-1) \times (\mathbf{q}-1)$

Find $\mathbf{d}$ so that $(\mathbf{e} \times \mathbf{d}) \bmod ((\mathbf{p}-1) \times (\mathbf{q}-1)) = 1$

Modular inverse using extended Euclidean algorithm

Define $\mathbf{N} = \mathbf{p} \times \mathbf{q}$

RSA

Alice picks

- $\mathbf{p}$ and $\mathbf{q}$ as large, random, k-bit prime numbers
- $\mathbf{e}$ as relatively prime to $(\mathbf{p}-1) \times (\mathbf{q}-1)$

Find $\mathbf{d}$ so that $(\mathbf{e} \times \mathbf{d}) \bmod ((\mathbf{p}-1) \times (\mathbf{q}-1)) = 1$

Define $\mathbf{N} = \mathbf{p} \times \mathbf{q}$

Factoring out $\mathbf{p}$ and $\mathbf{q}$ is infeasible

RSA

Alice picks

- $\mathbf{p}$ and $\mathbf{q}$ as large, random, k-bit prime numbers
- $\mathbf{e}$ as relatively prime to $(\mathbf{p}-1) \times (\mathbf{q}-1)$

Find $\mathbf{d}$ so that $(\mathbf{e} \times \mathbf{d}) \bmod ((\mathbf{p}-1) \times (\mathbf{q}-1)) = 1$

Define $\mathbf{N} = \mathbf{p} \times \mathbf{q}$

Then $x^{\mathbf{de}} \bmod \mathbf{N} = x$ for any $x < \mathbf{N}$

RSA

Alice picks

- $\mathbf{p}$ and $\mathbf{q}$ as large, random, k-bit prime numbers
- $\mathbf{e}$ as relatively prime to $(\mathbf{p}-1) \times (\mathbf{q}-1)$

Find $\mathbf{d}$ so that $(\mathbf{e} \times \mathbf{d}) \bmod ((\mathbf{p}-1) \times (\mathbf{q}-1)) = 1$

Define $\mathbf{N} = \mathbf{p} \times \mathbf{q}$

Then $x^{\mathbf{de}} \bmod \mathbf{N} = x$ for any $x < \mathbf{N}$

Proof by Euler's theorem or Fermat's little theorem

RSA

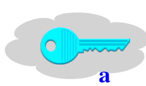
Alice picks

- **p** and **q** as large, random, k-bit prime numbers
- **e** as relatively prime to $(p\text{-}1) \times (q\text{-}1)$

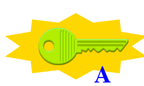
Find **d** so that $(e \times d) \bmod ((p\text{-}1) \times (q\text{-}1)) = 1$

Define $N = p \times q$

Then $x^{de} \bmod N = x$ for any $x < N$



$a = \langle d, N \rangle$



$A = \langle e, N \rangle$

RSA

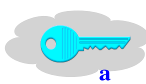
Alice picks

- $\mathbf{p}$ and $\mathbf{q}$ as large, random, k-bit prime numbers
- $\mathbf{e}$ as relatively prime to $(\mathbf{p}-1) \times (\mathbf{q}-1)$

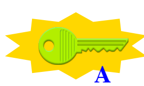
Find $\mathbf{d}$ so that $(\mathbf{e} \times \mathbf{d}) \bmod ((\mathbf{p}-1) \times (\mathbf{q}-1)) = 1$

Define $\mathbf{N} = \mathbf{p} \times \mathbf{q}$

Then $x^{\mathbf{de}} \bmod \mathbf{N} = x$ for any $x < \mathbf{N}$



$\mathbf{a} = \langle \mathbf{d}, \mathbf{N} \rangle$ $\qquad$ $\boxed{\text{plaintext}_i}^{\mathbf{e}} \bmod \mathbf{N} = \boxed{\text{ciphertext}_i}$



$\mathbf{A} = \langle \mathbf{e}, \mathbf{N} \rangle$ $\qquad$ $\boxed{\text{ciphertext}_i}^{\mathbf{d}} \bmod \mathbf{N} = \boxed{\text{plaintext}_i}$

RSA

Alice picks

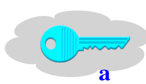
- $\mathbf{p}$ and $\mathbf{q}$ as large, random, k-bit prime numbers
- $\mathbf{e}$ as relatively prime to $(\mathbf{p}-1) \times (\mathbf{q}-1)$

Find $\mathbf{d}$ so that $(\mathbf{e} \times \mathbf{d}) \bmod ((\mathbf{p}-1) \times (\mathbf{q}-1)) = 1$

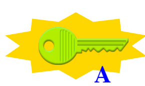
Define $\mathbf{N} = \mathbf{p} \times \mathbf{q}$

Then $x^{\mathbf{de}} \bmod \mathbf{N} = x$ f

k-bit chunk of message



$\mathbf{a} = \langle \mathbf{d}, \mathbf{N} \rangle$ $\quad$ $\boxed{\text{plaintext}_i}^{\mathbf{e}} \bmod \mathbf{N} = \boxed{\text{ciphertext}_i}$



$\mathbf{A} = \langle \mathbf{e}, \mathbf{N} \rangle$ $\quad$ $\boxed{\text{ciphertext}_i}^{\mathbf{d}} \bmod \mathbf{N} = \boxed{\text{plaintext}_i}$

RSA versus a Block Cipher

Compared to AES

- RSA is 1000x slower
- RSA has 10x larger keys (e.g., 2048 bits vs. 192 bits)
- RSA is more complex

... but RSA requires no initial shared secret

Using RSA

Generate a key pair:

```
openssl genrsa -out private.pem 1024
```

```
openssl rsa -pubout -in private.pem > public.pem
```

Sign a message:

```
openssl rsautl -sign -inkey private.pem -in a.txt > sig
```

Verify a signed message:

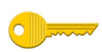
```
openssl rsautl -verify -pubin -inkey public.pem -in sig
```

Summary

Public key cryptography uses public information to bootstrap a private conversation

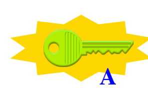
Diffie-Hellman

A way to arrive at a shared secret 

Shared  can then be used for a stream cipher, for example

Relies on the difficulty of the **discrete logarithm problem**

RSA

Published public key 

enables **confidential** message to owner,
authentication by owner

Relies on the difficulty of **prime factorization**