Public key cryptography
ACCESS 2012
based on Tom Davis' and Nick Korevaar's notes.

Before exchanging encrypted messages Alice and Bob do some preliminary work.

Alice Ⓐ

- Picks two large primes: p_A and q_A. (sssh it's a secret!)
- Computes modulus $N_A = p_A q_A$.
- Picks encryption power e_A such that

$$\gcd(e_A, (p_A - 1)(q_A - 1)) = 1.$$

Public key: N_A | e_A

Bob Ⓑ

- Picks two large primes: p_B and q_B. (sssh it's a secret!)
- Computes modulus $N_B = p_B q_B$.
- Picks encryption power e_B such that

$$\gcd(e_B, (p_B - 1)(q_B - 1)) = 1.$$

Public key: N_B | e_B

Scenario 1. Bob wants to send a secret MESSAGE to Alice.

Alice Ⓐ	Bob Ⓑ
	1. Ⓑ transcribes MESSAGE into an integer x (or several if MESSAGE is too long)
	2. Ⓑ encrypts message using Alice's public key: N_A | e_A
	$y = E_A(x) = x^{e_A} \mod N_A$
3. Ⓐ knows her number theory and p_A and q_A so she can find her decryption power d_A by solving the multiplicative inverse equation $e_A d_A \equiv 1 \mod (p_A - 1)(q_A - 1)$	
4. Ⓐ decrypts the message $x \equiv D_A\left(y\right) \equiv y^{d_A} \mod N_A.$	

The decryption function works because of Fermat's little theorem, indeed:

$$D_A(E_A(x)) \equiv D_A(x^{e_A}) \equiv (x^{e_A})^{d_A} \equiv x^{e_A d_A} \equiv x^{1+k(p_A-1)(q_A-1)} \equiv x \mod N_A.$$

Problem: Evil can send an message to Alice pretending to be Bob!

Scenario 2. Bob wants to send a secret MESSAGE to Alice with **secure signature**.

Alice Ⓐ	Bob Ⓑ
• Picks two large primes: p_A and q_A. (sssh it's a secret!) • Computes modulus $N_A = p_A q_A$. • Picks encryption power e_A such that $\gcd(e_A, (p_A - 1)(q_A - 1)) = 1.$ **Public key:** N_A e_A **Private key:** d_A (with $x^{e_A d_A} \equiv x \mod N_A$). **Signature:** $s_A \equiv$ integer(s) $< N_A$ transcribing to e.g. "signed by Alice".	• Picks two large primes: p_B and q_B. (sssh it's a secret!) • Computes modulus $N_B = p_B q_B$. • Picks encryption power e_B such that $\gcd(e_B, (p_B - 1)(q_B - 1)) = 1.$ **Public key:** N_B e_B **Private key:** d_B (with $x^{e_B d_B} \equiv x \mod N_B$). **Signature:** $s_B \equiv$ integer(s) $< N_B$ transcribing to e.g. "signed by Bob".

Alice Ⓐ	Bob Ⓑ
	1. Ⓑ **decrypts** his signature s_B with Ⓑ's private key $D_B(s_B).$
	2. Ⓑ appends message x to $D_B(s_B)$ creating $x\#D_B(s_B)$ (breaks this into blocks $< N_A$) and encrypts using Ⓐ's public key: $y = E_A(x\#D_B(s_B))$
	EVIL doesn't know D_B so EVIL • can't get to $x\#D_B(s_B)$ • can't read message x • can't forge messages to Ⓐ which look like they came from Ⓑ.
3. Ⓐ decodes message y: $D_A(y) = D_A(E_A(x\#D_B(s_B)))$ $= \underbrace{x}_{\text{message}} \# \underbrace{D_B(s_B)}_{\text{gibberish}}$	
4. Ⓐ uses Ⓑ's public key to compute: $E_B(D_B(s_B)) = s_B.$ and only Ⓑ could make $D_B(s_B)$!!	