DATA PRIVACY AND SECURITY

Prof. Daniele Venturi

Master's Degree in Data Science
Sapienza University of Rome

CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

CHAPTER 2: Asymmetric Cryptography

Number Theory

"Cubum autem in duos cubos, aut quadratoquadratum in duos quadratoquadratos, et generaliter nullam in infinitum ultra quadratum potestatem in duos eiusdem nominis fas est divider cuius rei demonstrationem mirabilem sane detexi. Hanc marginis exiguitas non caperet."—Fermat's Last Theorem

Modular Arithmetic

- Quotient, reminder and gcd

$$a \bmod n = r \Rightarrow a = nq + r$$

  - E.g., $2 \bmod 7 = 2; 8 \bmod 7 = 1$
- **Congruences:** $a \equiv b \bmod n$ if n divides $a - b$
- $(\mathbb{Z}_n, +, \cdot)$ is a **ring**
  - If $\gcd(a, n) > 1$, then a **not invertible**
  - $\varphi(n) = \#\{a < n$ and co-prime with $n\}$
- $(\mathbb{Z}_p, +, \cdot)$ is a **field** (p is a prime)
  - $\mathbb{Z}_p^* = \{1, \dots, p-1\}$; $\exists g$ a **generator**

Euclidean Algorithm

- **Lemma:** For all $a \geq b > 0$, $\gcd(a, b) = \gcd(b, a \bmod b)$
- **Theorem:** For all $a \geq b > 0$, we can find u, v such that $\gcd(a, b) = au + bv$
- Example: Take $a = 14$ and $b = 10$
  - $14 = 1 \cdot 10 + 4; 10 = 2 \cdot 4 + 2; 4 = 2 \cdot 2 + 0$ and in fact $\gcd(14{,}10) = 2$

$$2 = 10 - 2 \cdot 4 = 10 - 2(14 - 1 \cdot 10) = -2 \cdot 14 + 3 \cdot 10$$
$$\Rightarrow (u, v) = (-2{,}3)$$

Basic Facts

- **Euler's Theorem:** Let $n > 0$. For all $a \in \mathbb{Z}_n^*$:
$$a^{\varphi(n)} \equiv 1 \bmod n$$
- **Corollary:** For a prime p and all a such that $p \nmid a$, we have $a^{p-1} \equiv 1 \bmod p$
- Example: Take $\mathbb{Z}_{10}^* = \{1,3,7,9\}$
  – Note that $\varphi(10) = 4$
  – 3 is a generator: $3^0 \equiv 1, 3^1 \equiv 3, 3^2 \equiv 9, 3^3 \equiv 7$
  – For $a = 7$, we have $7^4 \equiv 1 \bmod 10$

Primality Testing

- Every integer n is either a prime or it can be written as a product of primes (Euclid)
  - Such a prime decomposition is unique (Gauss)
- There are **infinitely many** primes (Euclid)
  - For large n there are $\approx \frac{n}{\ln(n)}$ primes in $[n]$ (PNT)
- We can **efficiently test** if an integer is a prime
  - Thanks to a famous algorithm by Agrawal, Kayal, and Saxena

Fermat's Test

- Given a value p to test, pick a random a not divisible by p and check if $a^{p-1} \equiv 1 \bmod p$
  - If not, conclude p is **composite**
  - If yes, conclude p is **probably prime**
- Let a be s.t. $a^{n-1} \equiv 1 \bmod n$ for composite n
  - a is a **Fermat liar**, and n is a **Fermat pseudoprime**
  - There are **infinitely** many Fermat pseudoprimes
  - There are **infinitely** many Carmichael numbers, i.e. numbers n for which **all** values of a co-prime with n are Fermat liars

Integer Factoring

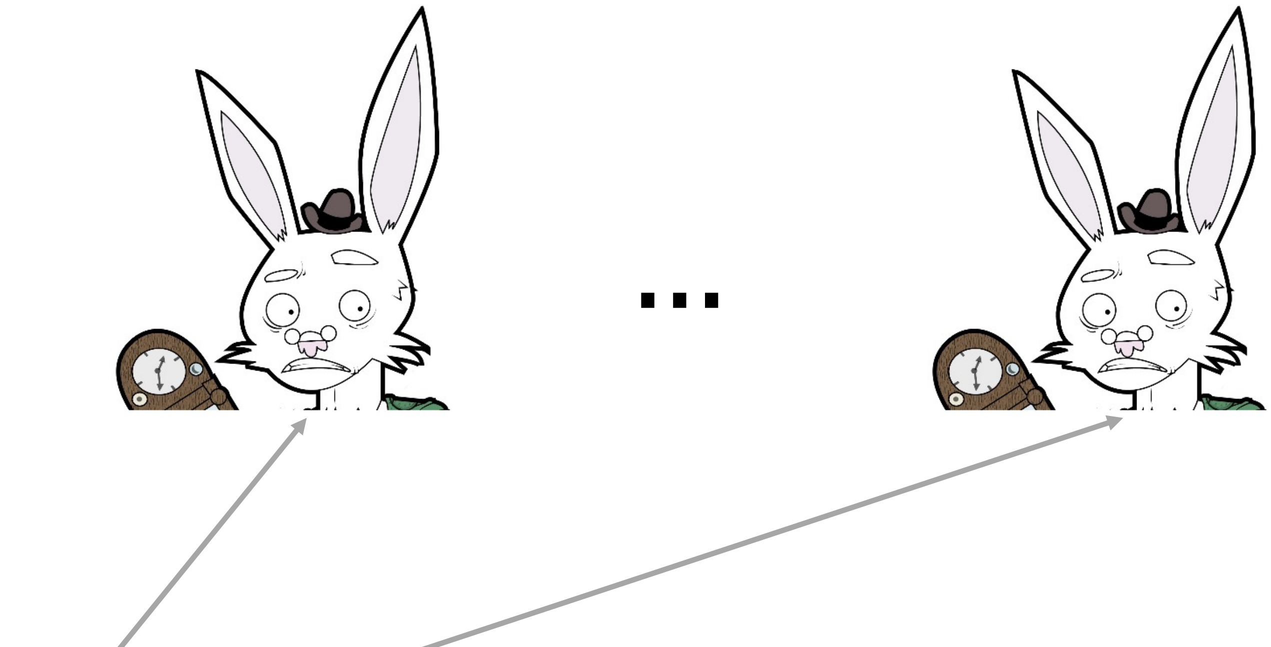
- Let $n = p \cdot q$. Goal: Given n, find p, q
- Brute force: Divide n for all values $\leq \sqrt{n}$
  - Complexity $O(\log^2(n) \cdot p/\ln(p))$ is **exponential** in n whenever $p \approx \sqrt{n}$
- Many attempts and algorithms
  - Pollard, Quadratic and Number Field Sieve
  - Complexity is sub-exponential in n
- RSA challenges
  - Last challenge (RSA-768) took over 2 years on a **huge computer network**

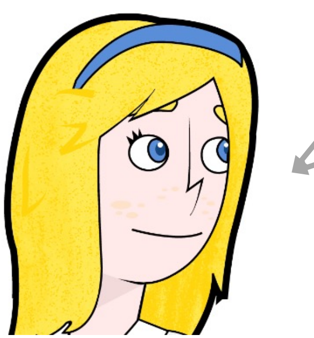
CIS SAPIENZA
RESEARCH CENTER FOR CYBER INTELLIGENCE AND INFORMATION SECURITY

Discrete Logarithm

- Let $\mathbb{Z}_p^* = \{1,2, \dots, p-1\}$, for a prime p
- For each $y \in \mathbb{Z}_p^*$, $y = g^x$ for some x
- **Discrete Log assumption:** Given (y, g, p) compute x
  - I.e., modular exponentiation is a OWF
- Deeply studied problem
  - Best algorithms have complexity **sub-exponential** in the size of p

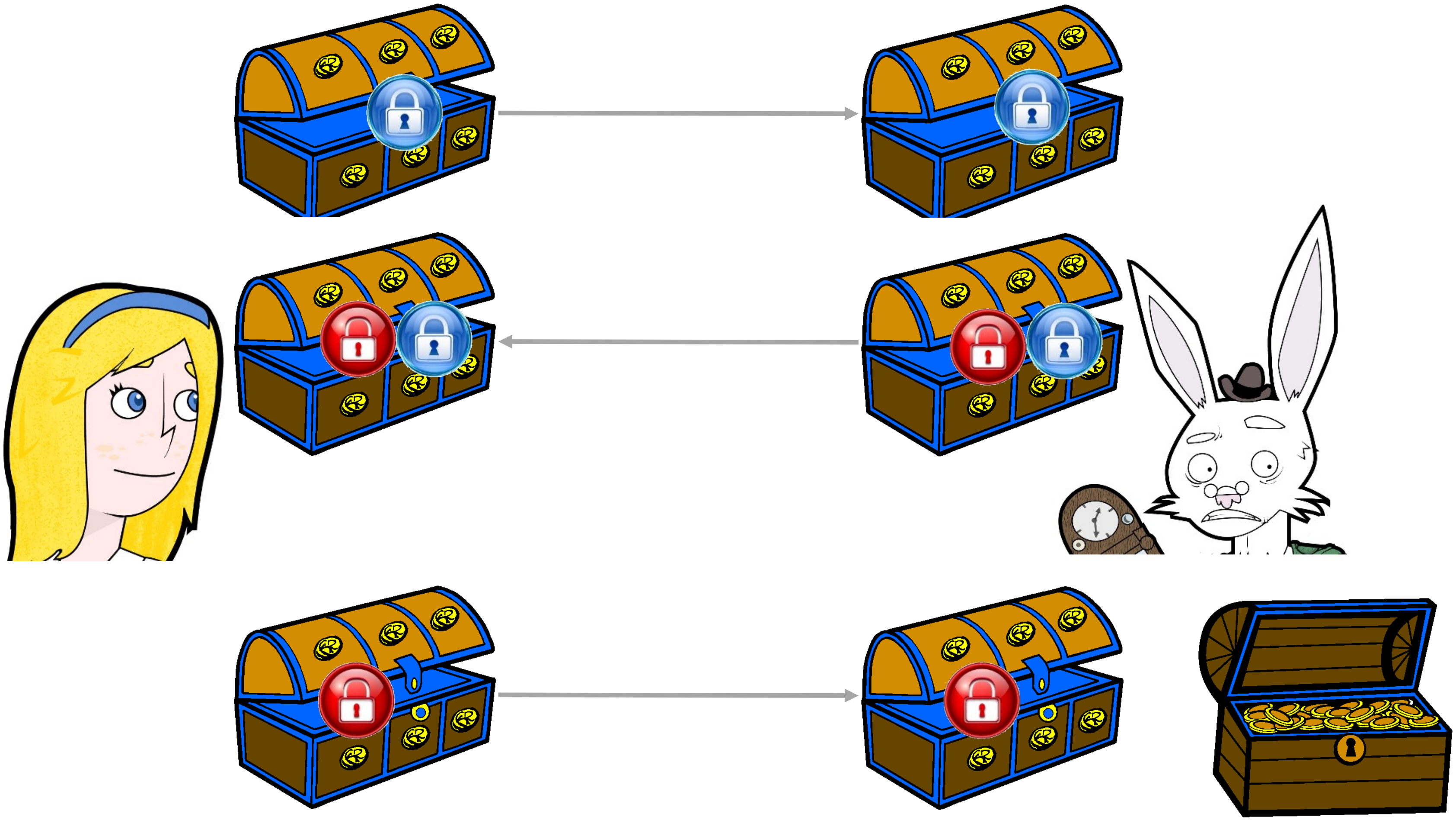
The Key Distribution Problem





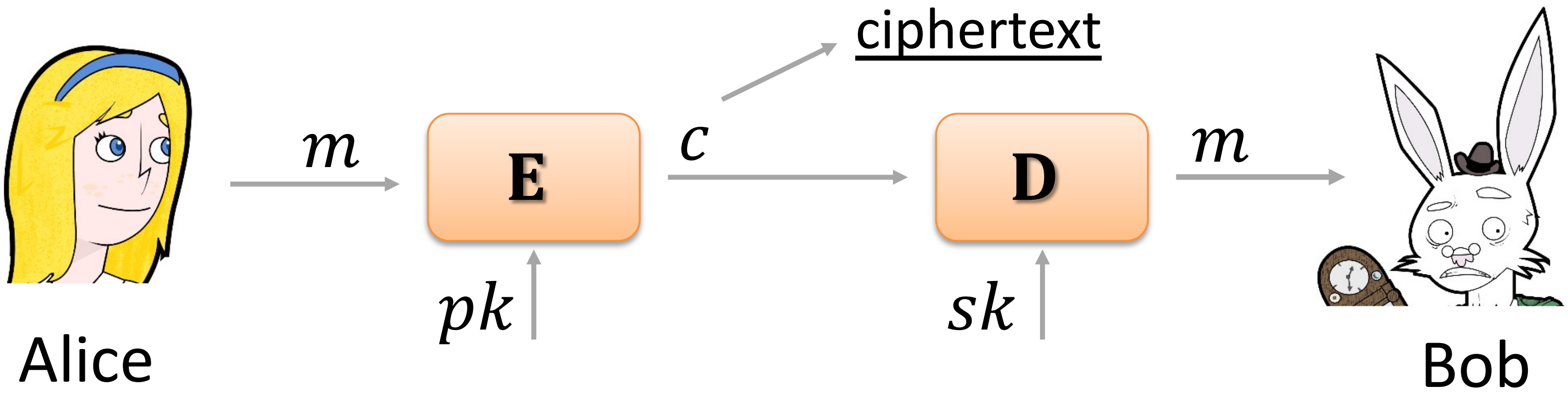
- Too many keys: $O(n^2)$
- Hard to **distribute** them
- Hard to **store** them

The Public Key Revolution



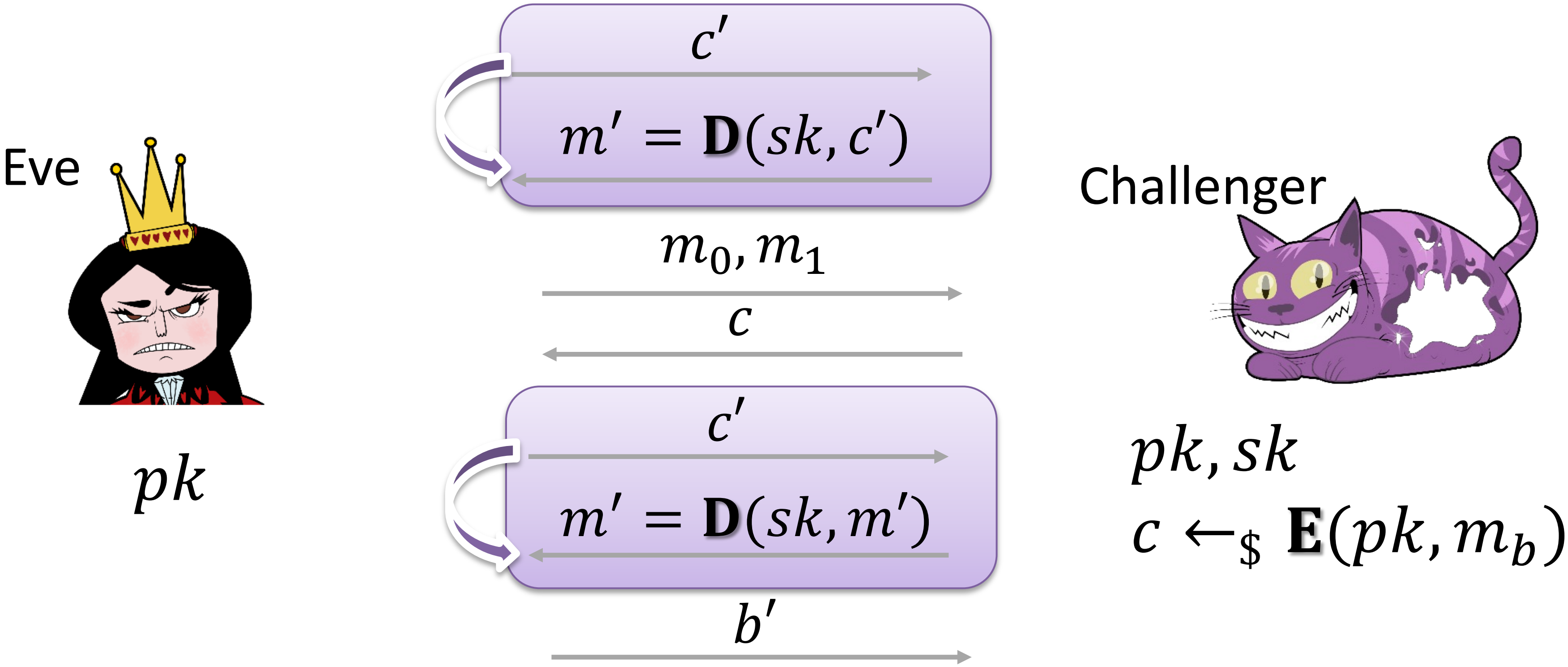
Public-Key Encryption

- Bob has a key pair (pk, sk)



- Must be **infeasible** to compute sk from pk
- No PKE scheme can achieve **unconditional security**

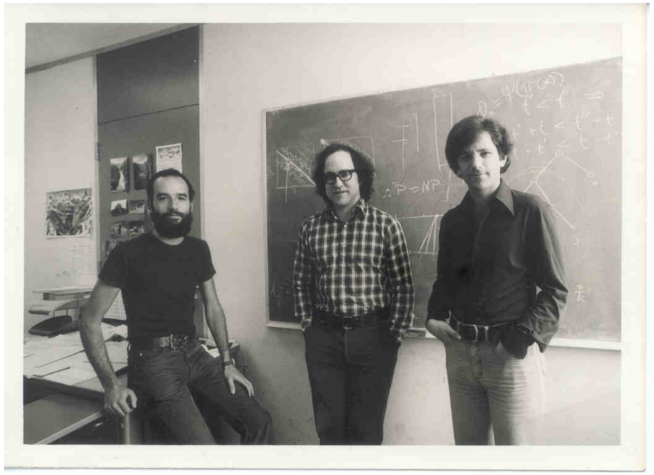
CPA/CCA Security



- No encryption queries (implicit given pk)
  - Cannot query on **challenge ciphertext**
- CCA security captures **non-malleability**

Hystory of PKE

- Concept **proposed** by Diffie & Hellman (1976)
- Rivest, Shamir, Adleman invent RSA (1978)
  - Very similar idea proposed by James Ellis in 1970 while working for GCHQ (but **top secret**)
- CPA security (Goldwasser and Micali – 1984)



Key Encapsulation

- PKE is one of the most significant advance in the 3000 years history of cryptography
- **Complementary** rathen than a replacement of secret-key cryptography
  - PKE algorithms are **expensive**
  - Use PKE to share a secret (session) key and later encrypt communication with AES
  - Idea used in the Transport Layer Security (TLS) protocol (more on this later)

Textbook RSA

- Let $n = p \cdot q$ and e, d s.t. $e \cdot d \equiv 1 \bmod \varphi(n)$
- Set $pk = (n, e)$ and $sk = (d, p, q)$
- Encryption of $m \in \mathbb{Z}_n^*$: Return $c = m^e \bmod n$
- Decryption of $c \in \mathbb{Z}_n^*$: Return $c^d \bmod n$
  - **Correctness:** $c^d \equiv m^{ed} \equiv m$ (by Euler's Theorem)
- Parameters generation
  - Need to sample **large primes** (primality testing)
  - Sample e at random and compute the inverse of e modulo $\varphi(n)$ (using the **Euclidean algorithm**)

Remarks

- Need to **encode** bits in $\mathbb{Z}_n^*$
- Efficiency
  - Modular exponentiation: **"Square and multiply"**
  - Speed up using tricks from number theory
  - Small e makes encryption faster and $|pk|$ smaller
  - Harder to test for primality and need $|m| > |n|/3$
- Ciphertexts are **malleable**!
  - Given (m_1, c_1) and (m_2, c_2), then $c_1 \cdot c_2$ is an encryption of $m_1 \cdot m_2$

RSA with Padding

- Clearly Textbook RSA is not CPA secure (why?)
- **Randomized** version
  - Encrypt $r||m$ (for random r)
  - Discard r on decryption
- PKCS Standard

$$0||2||r\text{ (at least 8 bytes)}||0||m$$

  - First byte s.t. the obtained integer is $< n$
  - Second byte encodes mode (i.e., encryption) and enforces **modular reduction**

The RSA Assumption

- Given $y = x^e$ (for $x \leftarrow_\$ \mathbb{Z}_n^*$), compute x
- I.e., compute the **e-th root** modulo $n = p \cdot q$
- RSA **implies** Factoring
  - If one can factor n, it can also compute $\varphi(n)$ and thus d
- Other direction **not known**
  - But best algorithm for breaking RSA requires factoring the modulus

ElGamal PKE

- Let $h = g^x$ for g a **generator** of $\mathbb{Z}_p^*$ and random x
- Set $pk = (g, p, h)$ and $sk = x$
- Encryption of $m \in \mathbb{Z}_p^*$: Pick **random** r and return $c = (c_1, c_2) = (g^r, h^r \cdot m)$
- Decryption of $(c_1, c_2) \in (\mathbb{Z}_p^*)^2$: Return c_2 / c_1^x
  - $\frac{c_2}{c_1^x} = \frac{h^r \cdot m}{(g^r)^x} = \frac{h^r \cdot m}{(g^x)^r} = \frac{h^r \cdot m}{h^r} = m$

The CDH Assumption

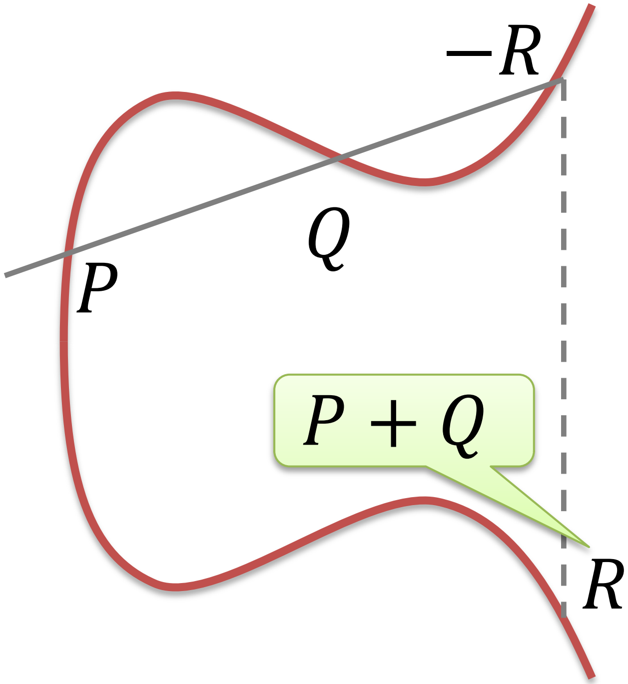
- Let $\mathbb{Z}_p^* = \{1,2,\ldots,p-1\}$, for a prime p and let g be a **generator**
- Given (g, p, g^x, g^y) for random x, y, **compute** g^{xy}
- CDH **implies** DL
  - If we can solve DL, we can compute x (or y) and thus obtain $g^{xy} = (g^y)^x$ (or $g^{xy} = (g^x)^y$)
- Other direction **not known**
  - But best algorithm for solving CDH requires to compute a DL

The DDH Assumption

- **Distinguish** (g, p, g^x, g^y, g^z) for random x, y, z from (g, p, g^x, g^y, g^{xy})
- DDH **implies** CDH
  - If we can solve CDH, we can compute g^{xy} and thus distinguish between g^{xy} and g^z
- Other direction is **false**
  - Simply because DDH **does not hold** in $\mathbb{Z}_p^*$
  - Take $p = 2q + 1$ (for primes p, q) and let $\mathbb{G}$ be the subgroup of $\mathbb{Z}_p^*$ consisting of all elements $y = x^2$
  - The order of $\mathbb{G}$ is q

Elliptic-Curve Cryptography

- The points of a curve $E: Y^2 = X^3 + aX + b$ modulo a prime p form a **group** $\mathbb{G} = (E, +)$



- Discrete logarithm: Given $Q = xP$, find x
  - DDH believed to be hard
- **Bilinear map** $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that

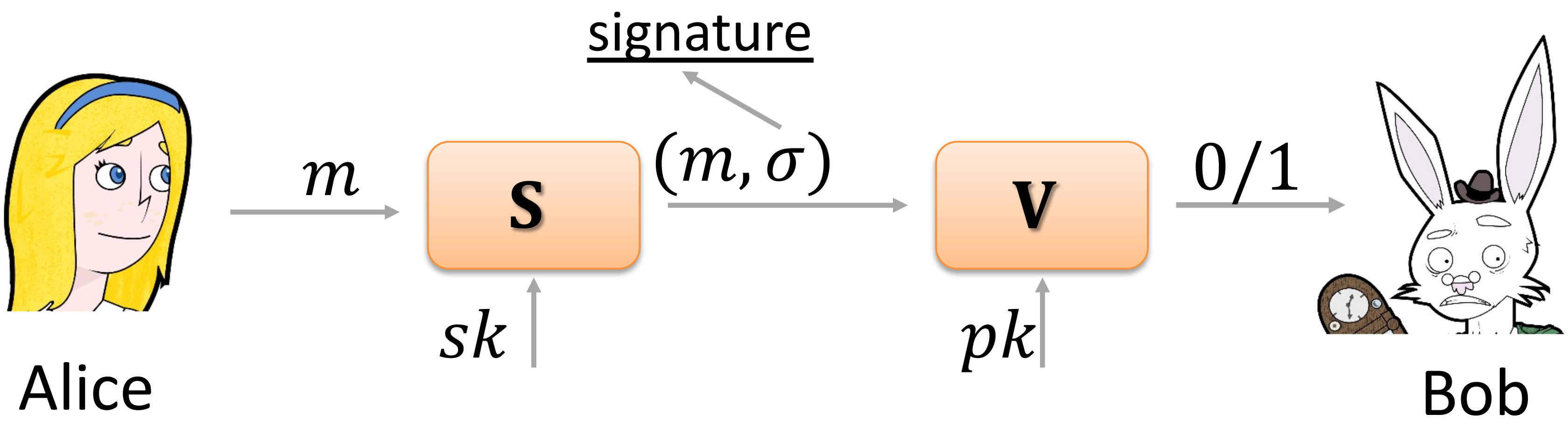
$$\forall a, b: e\left(g^a, g^b\right) = e(g, g)^{ab}$$

The Bilinear Diffie-Hellman Assumption

- When $\mathbb{G}$ is a **bilinear group**, DDH is easy in $\mathbb{G}$
  - Given (g, p, e, X, Y, Z) can simply check that $e(X, Y) = e(g, Z)$
- However, the following **variant of CDH** is believed to hold:
  - Given (g, p, e, g^x, g^y, g^z) compute $e(g, g)^{xyz}$
  - Once again BCDH **implies** CDH, but the other direction is **unknown**
- One can also assume DDH to be **hard** in $\mathbb{G}_T$
  - This is called the **SXDH assumption**

Digital Signatures

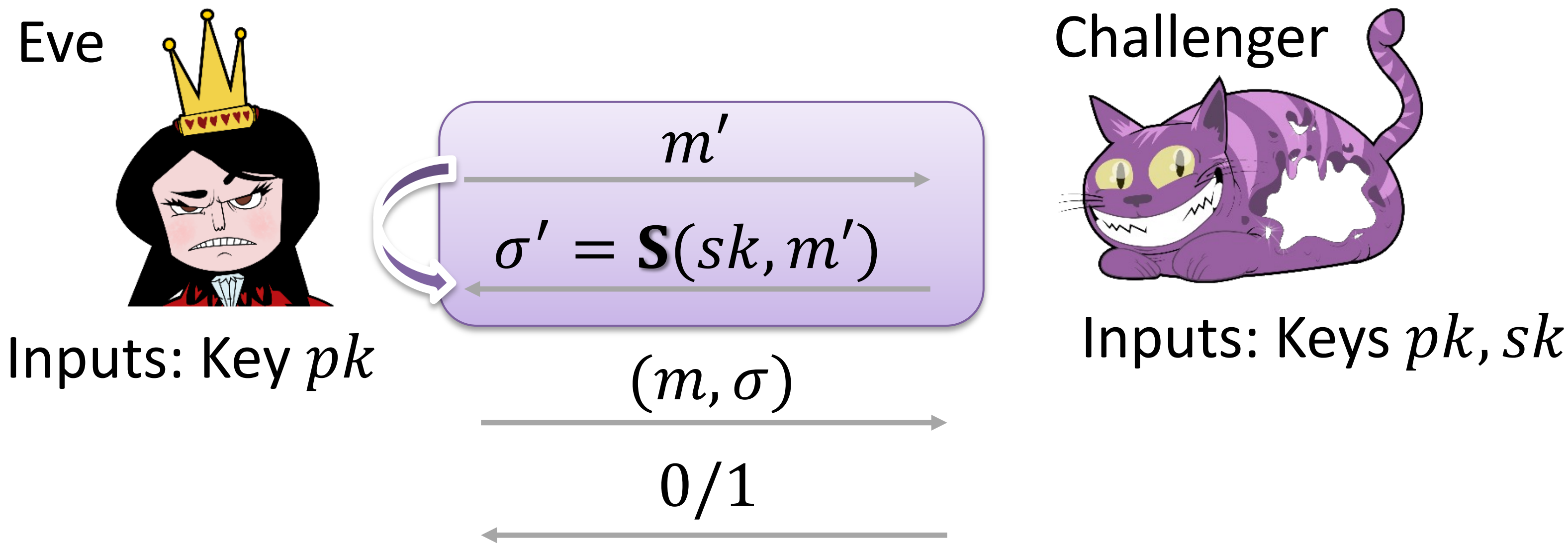
- Alice has a key pair (pk, sk)



- **Correctness:**
  $\forall pk, sk, m\colon \mathbf{V}(pk, (m, \mathbf{S}(sk, m))) = 1$
- **Security:** Should be hard to forge a signature on a message without knowing the secret key

CIS SAPIENZA
Research Center for Cyber Intelligence and Information Security

Unforgeability



- Adversary wins iff (m, σ) is **valid** and m is **fresh** (i.e. not asked during signing queries)

How to Sign with RSA

- Let $n = p \cdot q$ and e, d s.t. $e \cdot d \equiv 1 \bmod \varphi(n)$
- Set $pk = (n, e)$ and $sk = (d, p, q)$
- Signature of $m \in \mathbb{Z}_n^*$: Return $\sigma = m^d \bmod n$
- Verification of $(m, \sigma) \in \mathbb{Z}_n^* \times \mathbb{Z}_n^*$: Return 1 iff $\sigma^e = m \bmod n$
  - **Correctness:** $\sigma^e \equiv m^{ed} \equiv m$ (by **Euler's Theorem**)
- **Not secure!**
  - Pick any $\sigma \in \mathbb{Z}_n^*$ and output (m, σ) where $m = \sigma^e \bmod n$

Full-Domain Hash

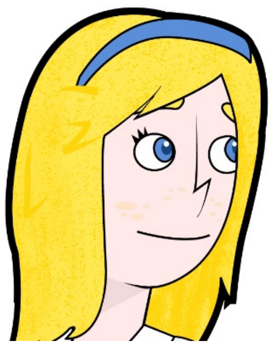
- Solution: First **hash** the message!
- Now, $\sigma = (\mathbf{H}(m))^d$ where $\mathbf{H}$ is a cryptographic hash function
- Possible attacks:
  - Pick any σ and let $\mathbf{H}(m) = \sigma^e$; hard to compute m from $\mathbf{H}(m)$ (**one-wayness**)
  - Given valid (m, σ), hard to find $m' \neq m$ s.t. $\mathbf{H}(m') = \mathbf{H}(m)$ (**weak collision resistance**)
  - Find $m \neq m'$ s.t. $\mathbf{H}(m) = \mathbf{H}(m')$ and obtain a valid signature on m (**strong collision resistance**)

Public-Key Infrastructure

- Need to **certify** public keys
  - Otherwise simple **man-in-the-middle** attacks are possible
- Have a Certification Authority (CA) confirm the **authenticity** of public keys
  - CA signs binding between identity and public key
- Single CA not a good solution
  - Unique point of failure
  - In practice: **Chains** of certificates

Basic Idea

 $_{\mathrm{CA} \to \mathrm{A}} \leftarrow_{\$} \mathbf{S}(sk_{\mathrm{CA}}, pk_{\mathrm{A}} || \mathrm{Alice})$

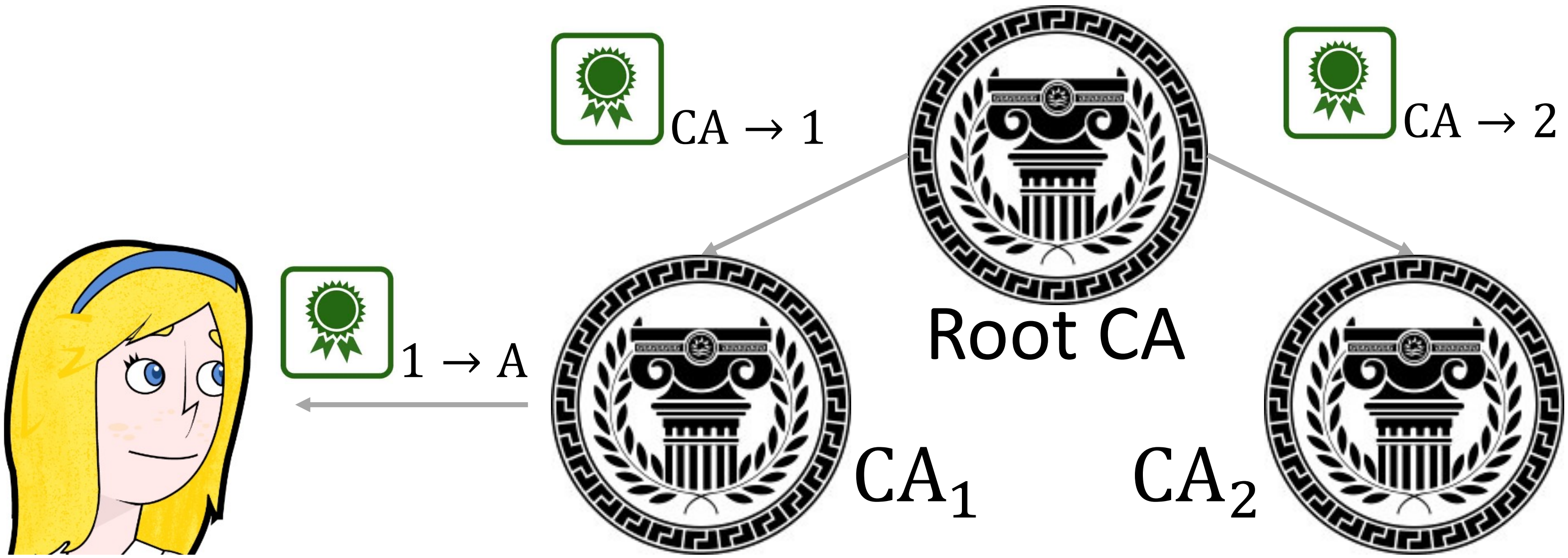


Alice, pk_{A}

 $_{\mathrm{CA} \to \mathrm{A}}$

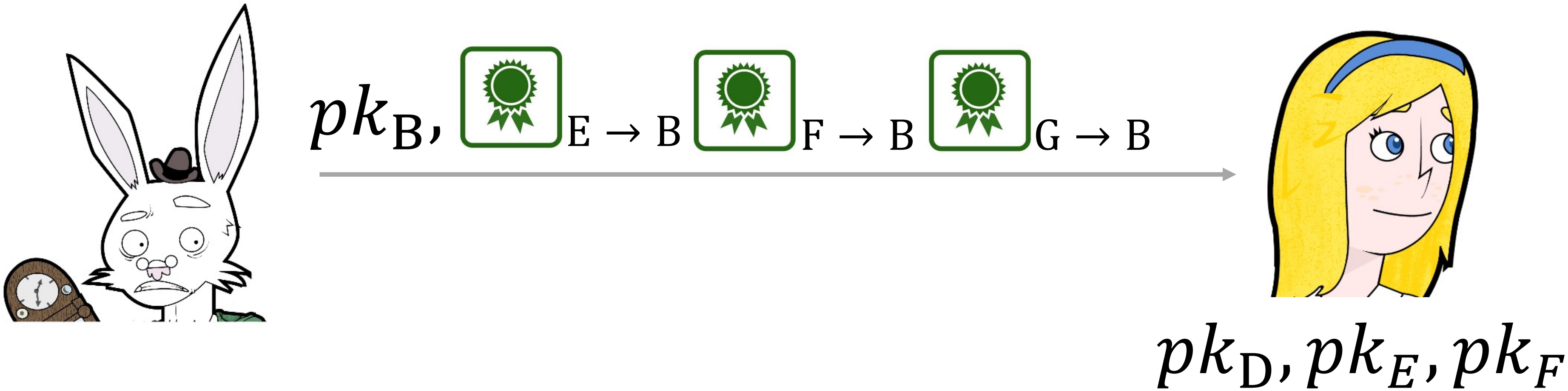
- Format of certificates **standardized** by ITU (X.509)
- Anybody can verify  $_{\mathrm{CA} \to \mathrm{A}}$ using pk_{CA}

Chain of Certificates



- Propagation of trust
- Each user might be a member of **different** PKIs
- Typically the binding involves credentials
  - User name only for management and auditing

Web of Trust



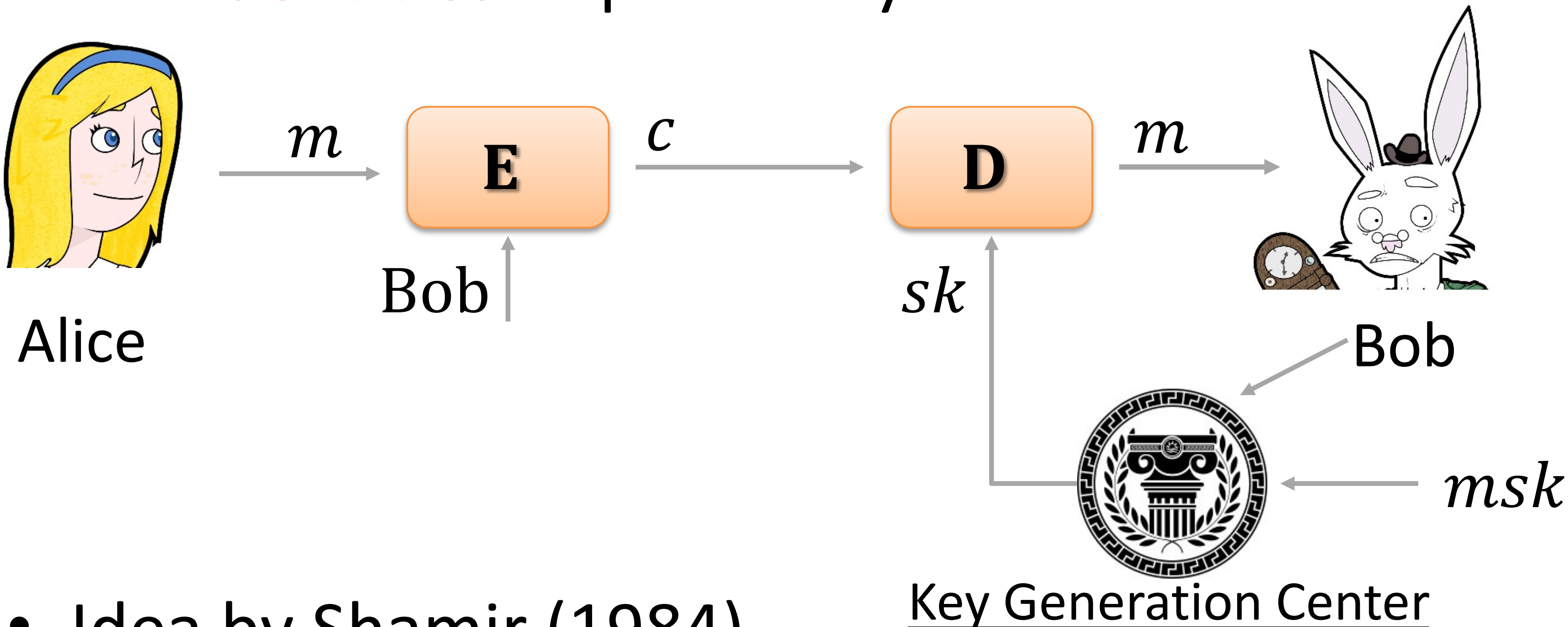
- Users can **self-certify** public keys
  - No trusted party required (fully distributed environment)
  - Approach taken in PGP (IETF)

Management of Certificates

- Certificates should have associated a **validity interval**
  - Need to check it didn't expire
  - What granularity?
- **Revocation** of certificates
  - Associate a different serial number to each certificate
  - Maintain a Certificate Revocation List
- CA must be **online** in order to answer validation queries

Identity-Based Encryption (IBE)

- Use **identities** as public keys

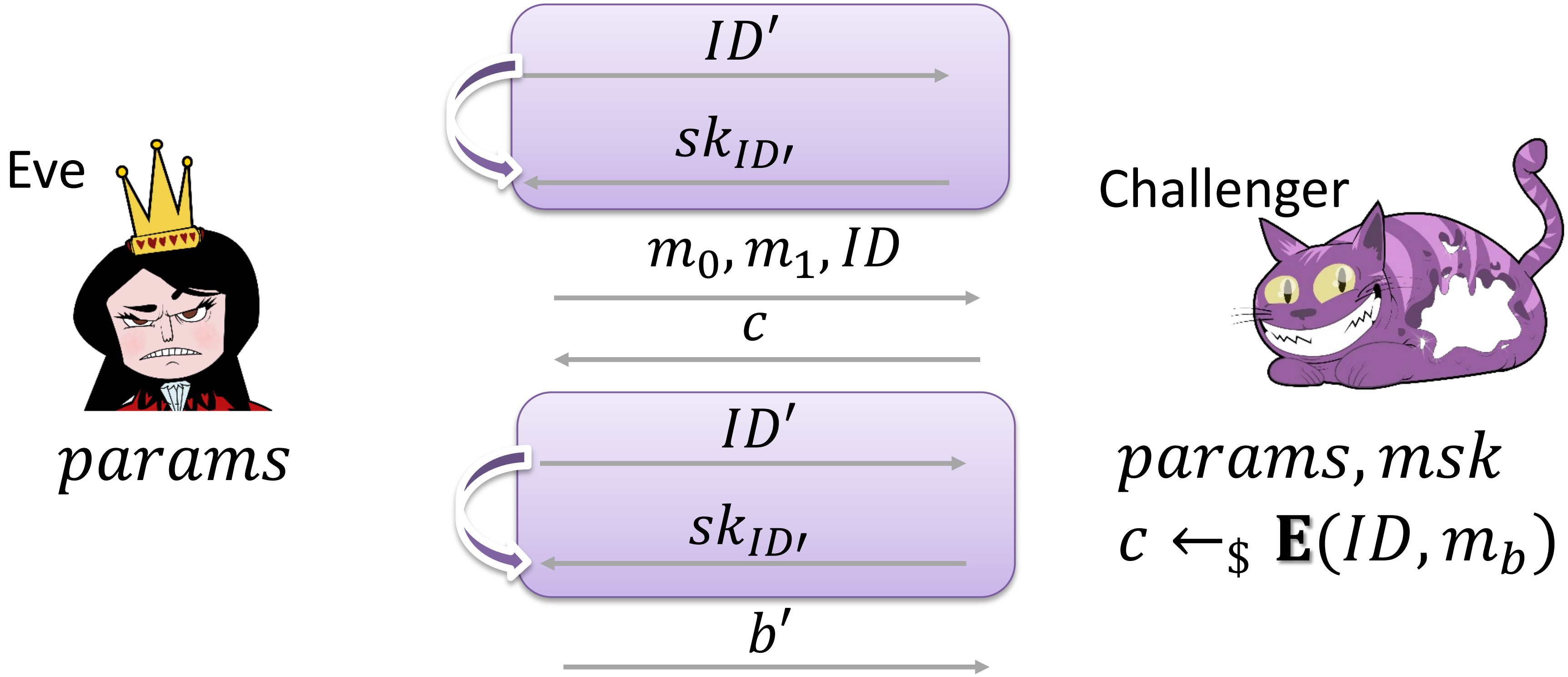


- Idea by Shamir (1984)
  - First realization by Boneh and Franklin (2001)

IBE vs. PKI

- Keys are **unrevocable** in IBE
  – Generate keys w.r.t. identity plus some timestamp
- PKI **not necessary** in IBE
- KGC is a high-value target to adversaries
  – Several mitigations are possible
- **Key escrow** (not present in PKI)
- IBE requires a **secure channel** in order to transmit the secret key to the user

Semantic Security of IBE

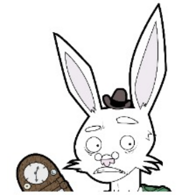


- Adversary **not allowed** to extract the secret key for to the **challenge identity** ID

Boneh-Franklin IBE

Pick random r

$$y_{ID} = e(\mathbf{H}_1(ID), h)$$



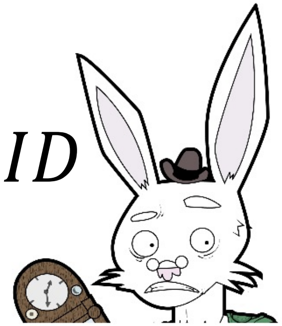
ID

$$d_{ID} = (\mathbf{H}_1(ID))^x$$

$$msk = x$$

$$(g, h = g^x)$$

$$c = (u, v) = (g^r, m \oplus \mathbf{H}_2(y_{ID}^r))$$



ID

$$e(d_{ID}, u) = e((\mathbf{H}_1(ID))^x, g^r)$$
$$= e((\mathbf{H}_1(ID), g)^{xr} = y_{ID}^r$$

$$m = v \oplus \mathbf{H}_2(e(d_{ID}, u)))$$