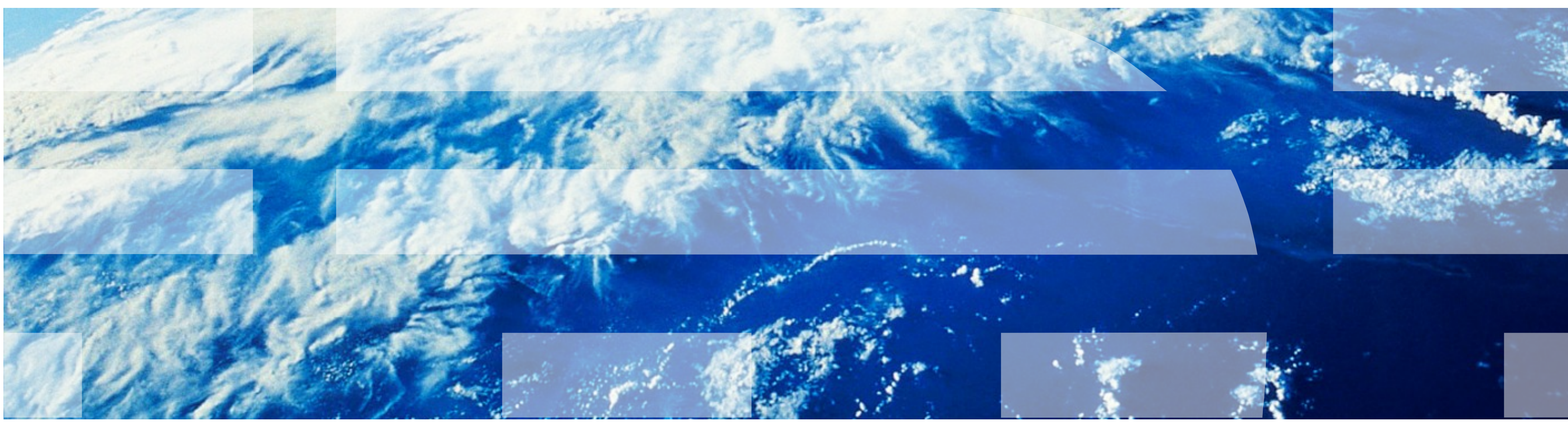
Alessandro Sorniotti

Distributed Cryptography



Overview

Distributed Crypto

- Used to distribute the ability to perform crypto operations among *n* parties s.t.
  - Any *t+1* parties can perform the operation
  - *t* parties cannot (provably) perform the operation
    - So up to *t* parties can be malicious or compromised
- Mostly asymmetric crypto operations

Secret Sharing

- Introduced by Shamir
- Algorithm that allows a dealer to share a secret *s* among *n* parties s.t.
  - Any *t+1* parties can recompute *s*
  - Any *t* parties cannot learn absolutely anything about *s*

Secret Sharing (cont'd)

- Secret Sharing; 2 protocols: *Share* and *Recover*
- *Share*
  - Trusted dealer picks a random t-degree polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_t x^t$
  - $s = f(0)$ is the secret
  - The i-th party ($i = 0, \ldots, n-1$) receives $s_i = f(i)$
- *Recover*
  - $t+1$ parties can reconstruct s

$$s = f(0) = \sum_{i \in \mathcal{S}} \lambda^{\mathcal{S}}_{0,i}\, s_i$$

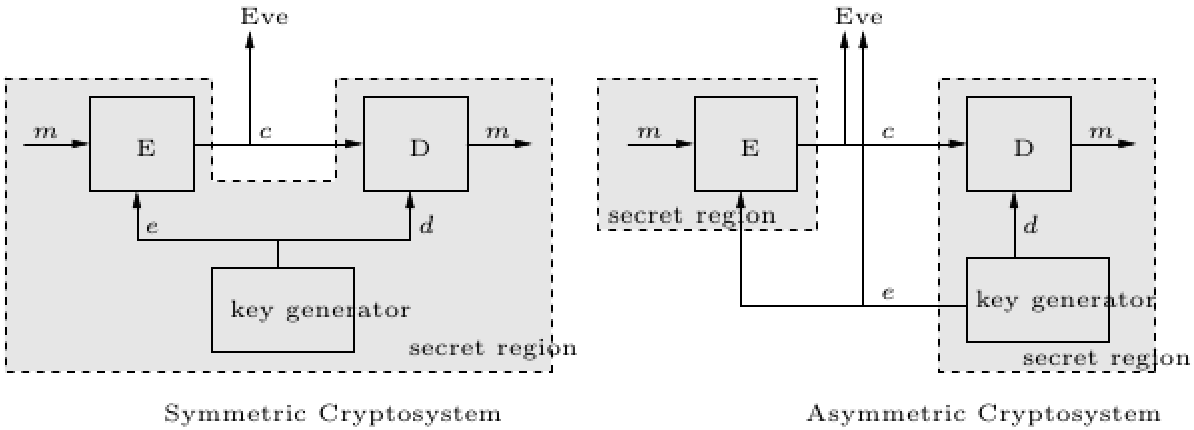
where $\lambda^{\mathcal{S}}_{0,i} = \prod_{j \in \mathcal{S}, j \neq i} \frac{j}{j-i}$

Secret Sharing (cont'd)

- Some observations
  - Scheme is information-theoretically secure
  - Some parties can be given more “power”
  - Can create complex access structures to a secret
  - Given *(t+1)*-out-of-*n* dealer can construct *(t+1)*-out-of-*m, m>n*
  - (less than t) malicious parties may still cause problems
    - VSS (other parties cannot lie about the value of their shares)
  - Dealer knows the secret
    - May be ok (e.g. company delegation)
  - What if I don't trust the dealer?
  - Shares are as large as the secret (good and bad)

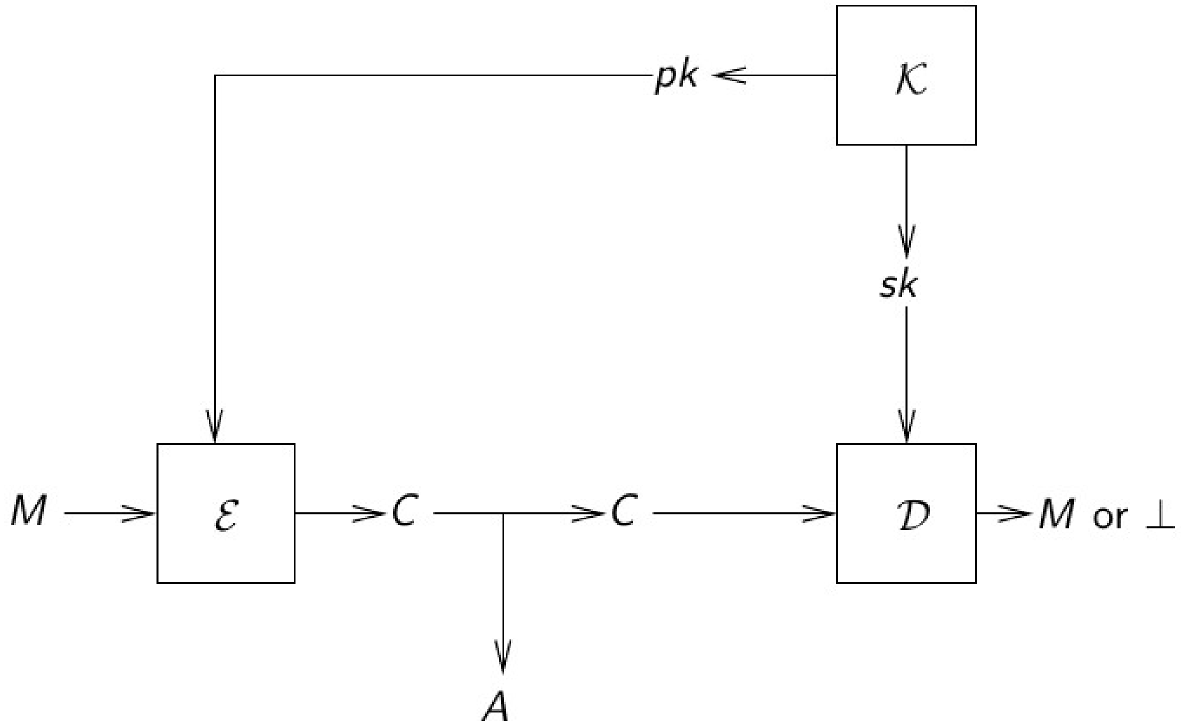
Public Key Encryption

- Symmetric encryption requires shared secret
- Asymmetric encryption “splits” the key in two
  - –Encryption (public) known to everyone
  - –Decryption (private) known to owner only



Public Key Encryption (cont'd)

- A public-key (or asymmetric) encryption scheme consists of three algorithms
- Requires
  - –Correctness
  - –Security



A quick recap on the security (cont'd)

- Security
  - Generally defined for crypto protocols as

Attacker breaks the scheme with negligible probability

- Who's the attacker?
- What is a negligible probability?
- What does it mean to “break” the scheme?

A quick recap on the security (cont'd)

Who's the attacker?

- Attacker modeled as a probabilistic poly-time Turing machine (p.p.t.)
  - Has access to randomness, can guess and be lucky ☺
  - Has limited resources (no information-theoretical security but computational)
    - Runs in $T(n) = n^c$ (poly-time, same for space)

A quick recap on the security (cont'd)

What is a negligible probability?

- Negligible function (given the security parameter *k*)

  Definition 3 (Negligible in terms of k ($\mathsf{negl}(k)$)) *An arbitrary function $v(k)$ (possibly a type of probability function) is* $\mathsf{negl}(k)$ *if:*

$$(\forall c > 0)\ (\exists k')\ (\forall k \geq k')\quad \left[v(k) \leq \frac{1}{k^c} \right]$$

- P(attacker breaks scheme) < *negl(k)*
  - Interested in the average case!
  - In practice, $P < 2^{-80}$ is considered secure today
  - $P < 2^{-64}$ is insecure
  - DES broken because key could be guessed with $P \sim 2^{-56}$
- Stays negligible if multiplied by any polynomial

A quick recap on the security (cont'd)

What should be "broken"?

- Attacker does not learn anything about the plaintext by seeing the ciphertext
  - Information-theoretical security;
    - Too strong!
- Whatever function the attacker can compute on the plaintext given the ciphertext, it can compute without it
  - Semantic security (computational)

A quick recap on the security (cont'd)

How do we prove semantic security?

- It's been proven identical to “testing” the attacker in the following way:

IND-CPA (Indistinguishability under chosen-plaintext attack)

- Attacker is given the public key
  - Can generate encryption of any message
- Attacker chooses two messages m_0 and m_1
- A fair coin *b* is flipped and $E(m_b)$ is given to the attacker
- The attacker guesses the value of *b*

A quick recap on the security (cont'd)

Putting it all together

"A public-key cryptosytem is semantically secure if any probabilistic poly-time Turing machine wins the IND-CPA game with negligible probability"

A quick recap on the security (cont'd)

Wait, but how do we prove that?

- By reduction to well-known "intractable problems"
  - Problems that are widely believed to be solved by p.p.t. Turing machine with negligible probability
- Examples

Problem	Given	Figure out
Discrete logarithm (DL)	g^x	x
Computational Diffie-Hellman (CDH)	g^x, g^y	g^{xy}
Decisional Diffie-Hellman (DDH)	g^x, g^y, g^z	Is $z \equiv xy \pmod{\lvert G \rvert}$?

where G is a particular cyclic group of prime order p; g is a generator of G; x, y and z are random integers in Z_p

A quick recap on the security (cont'd)

Wait, but how do we prove that?

- By reduction to well-known “intractable problems”
  - Cipher is built on intractable problem
  - Let's assume an attacker that can break the cipher exists
  - Then another p.p.t. Turing machine can “use” the attacker to solve the intractable problem
  - But the problem is intractable (by p.p.t. Turing machines)
  - Ergo attacker cannot exist

El-Gamal cryptosystem

- Three algorithms (K, E, D)
- $K(k)$
  - Pick "suitable" group G of prime order p and a generator g
  - Pick a random integer x in Z_p
  - Output $pk = \{G, p, g, y=g^x\}$; $sk = \{x\}$
- $E(m, pk)$
  - Pick a random integer r in Z_p and compute the "key" $K = y^r$
  - Output $c = (c_1, c_2) = (g^r, K \cdot m) = (g^r, g^{xr} \cdot m)$
- $D((c_1, c_2), sk)$
  - Compute $K = c_1^{\,x}$
  - Output $m = c_2 \cdot K^{-1}$

El-Gamal cryptosystem – security

- Intuitively, attacker cannot decrypt
  - Attacker doesn't know x
  - Needs to compute $K = g^{rx}$ from $c_1 = g^r$ and $y = g^x$
    - ~ breaking CDH
- Is that enough?
  - NO
  - Semantic security requires indistinguishability
  - Need to resort to DDH to prove semantic security

Theorem 1. *The above cryptosystem is polynomially secure under the DDH assumption.*

The proof, which is not presented in full detail here, is by hybrid argument: one proves that encryption of any message m is indistinguishable from a random pair (g^c, g^b). This follows easily from the DDH assumption. Therefore, encryptions of m_0 and m_1 are indistinguishable.

Threshold El-Gamal

- Share the power of decryption
- Three algorithms *(K, E, D)*
- *K(k)*
  - As before, except that $sk = \{x\}$ is now $(t+1)$-out-of-n secret-shared among n "decryptors" who receive $sk_i = \{x_i\}$
- *E(m, pk)* – unchanged
- $D((c_1, c_2), sk)$
  - Decryptors receive (c_1, c_2)
  - i-th decryptor computes a decryption share $d_i = c_1^{x_i}$
  - Given $t+1$ decryption shares, one can recompute $K = c_1^x$ and decrypt

$$\prod_{i \in \mathcal{S}} d_i^{\lambda^{\mathcal{S}}_{0,i}} = \prod_{i \in \mathcal{S}} c_1^{x_i \lambda^{\mathcal{S}}_{0,i}} = c_1^{\sum_{i \in \mathcal{S}} x_i \lambda^{\mathcal{S}}_{0,i}} = c_1^{x}$$

Threshold El-Gamal (cont'd)

- Some observations
  - Non-interactive
  - Use-cases
  - Dealer knows $sk = \{x\}$
  - Dealer can give more “power” to some decryptors
  - Given $(t+1)$-out-of-n dealer can construct $(t+1)$-out-of-m, $m>n$

Other “flavours” of threshold crypto

- Proactive schemes
  - Given (t+1)-out-of-n, t+1 parties can generate a new scheme which is (t'+1)-out-of-n'
  - Presently-untrusted parties are “left out”
- Verifiable schemes