CS-3002: Information Security

Lecture # 6: Public Key Encryption

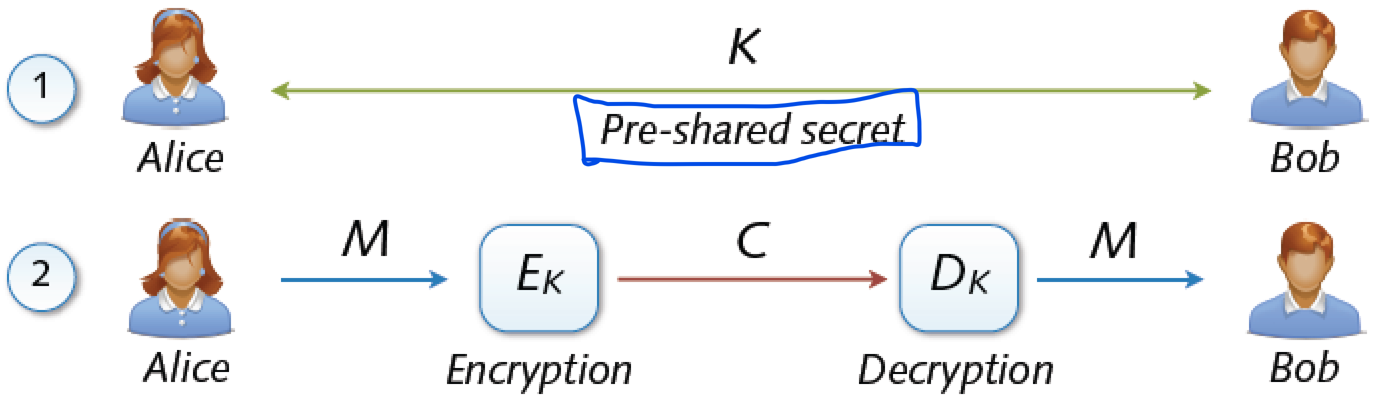
Prof. Dr. Sufian Hameed
Department of Computer Science
FAST-NUCES

Overview

- *What will you learn today*
  - *Public Key Encryption*
    - *Definition and Security*
    - *RSA Trapdoor*
    - *ISO Standard for RSA public key encryption*

Key Exchange

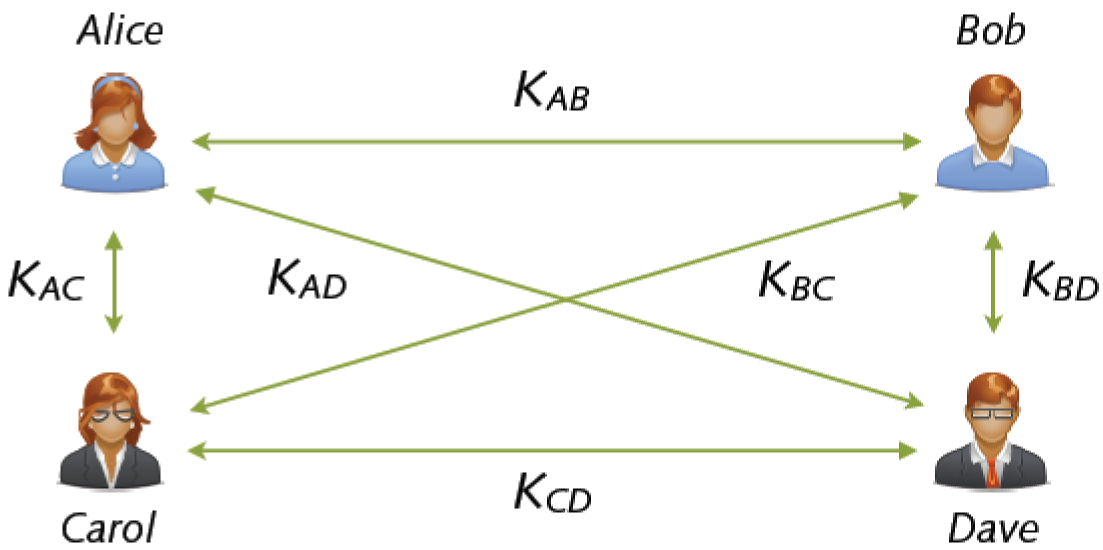
- **Symmetric cryptosystems secure and efficient, but ...**



- ***Precondition: secure exchange of keys in advance***
  - Paradox situation at a first glance
  - secure communication depends on secure key exchange

Multi-party Key Exchange

- **Involved multi-party key exchange with symmetric keys**
  - Quadratic growths: *n parties* → *(n2 - n) / 2 keys*



- **Problem rooted in symmetry (shared keys). *Alternatives?***

Asymmetric Keys

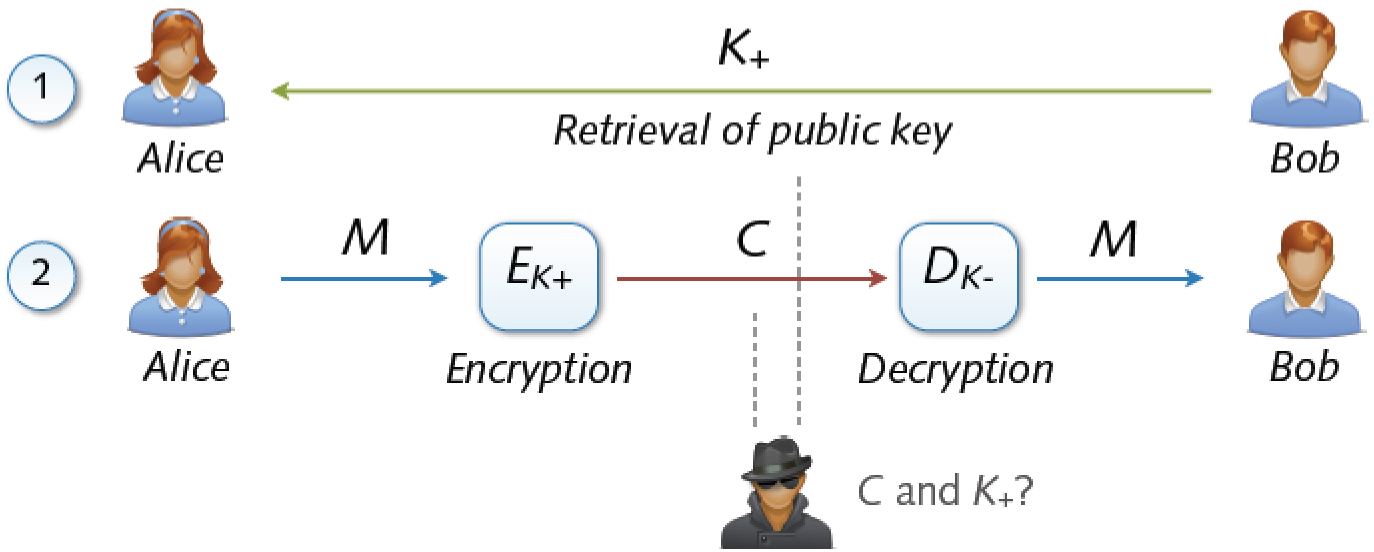
- **Solution: Two types of keys**
  - public key pk (K+) = enables encryption but no decryption
  - Private/secret key sk (K–) = used for decryption only
- Hard to deduce secret from public key

- **... similar to a classic mailbox**

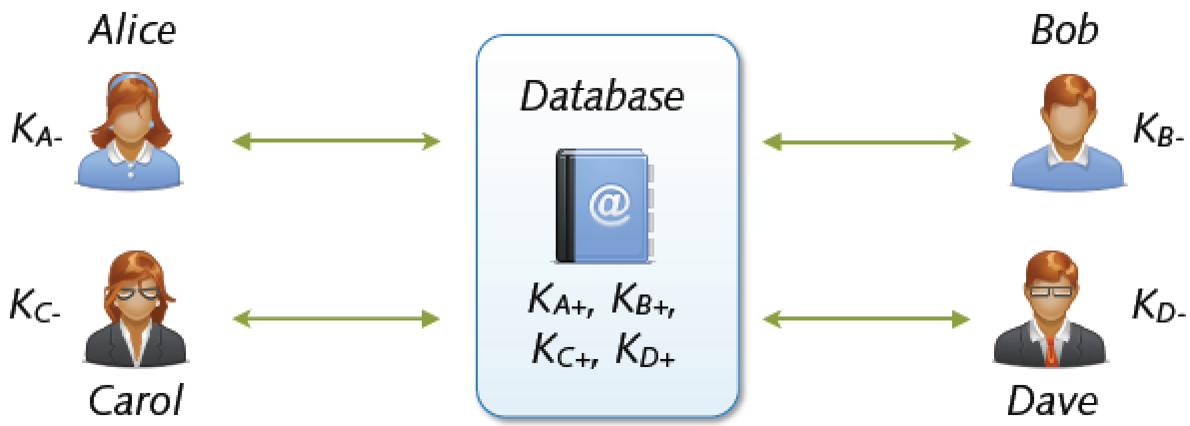
Asymmetric Cryptosystem

- **Asymmetric cryptosystems**
  - Asymmetric encryption and decryption
  - K+ (pk) = public key of Bob K– (sk) = secret key of Bob
  - No secure key exchange necessary

pk sk



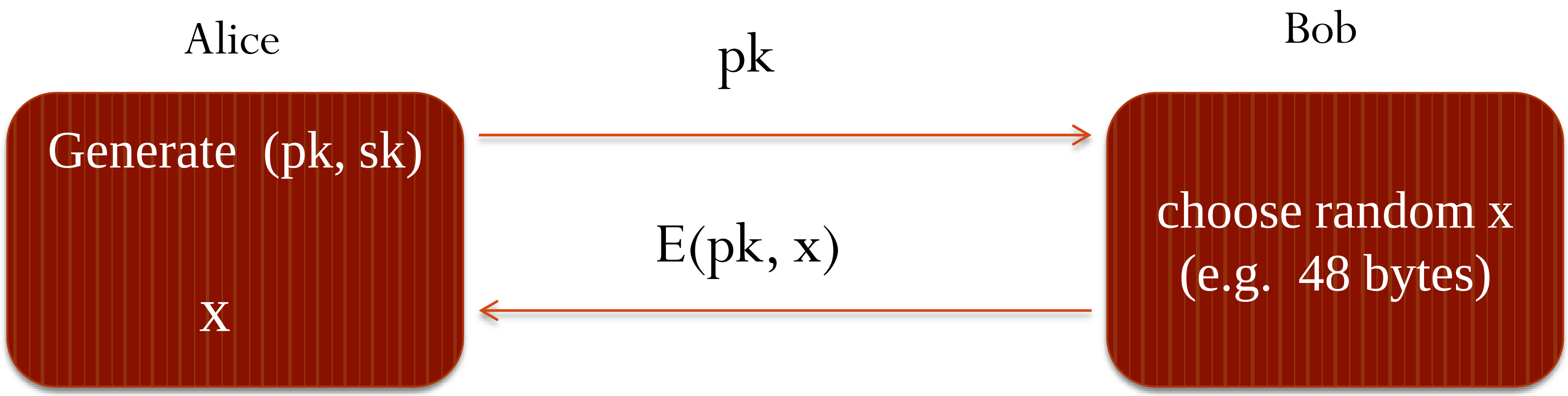
Key Exchange with Public Keys



- **Scalable communication with multiple parties**
  - Linear number of exchanges: n parties → n public keys
  - Real-world systems with millions of keys (e.g. PGP)
  - ... for the moment everything is fine

Applications

Session setup (for now, only eavesdropping security)



Non-interactive applications: (e.g. Email)

- Bob sends email to Alice encrypted using pk_{alice}
- Note: Bob needs pk_{alice} (public key management)

Hard Problems

- **Integer factorization**

Given an integer n, find its m prime factors

$$n = p_1 \cdot p_2 \cdots p_l \quad \text{with} \quad p_i \in \mathbb{P}$$

Examples: $12 = 2 \cdot 2 \cdot 3$ and $4711 = 7 \cdot 673$

- **Discrete logarithm**

Given integers a and b, find exponent x such that

$$a^x \equiv b \pmod{n} \quad \text{with} \quad x \in \mathbb{N}$$

Example: $2^6 \equiv 1 \pmod{7}$

- **Hardness: No polynomial-time algorithms known yet**

Trapdoor One-way Functions

- **One-way function $F(x) = y$ based on hard problem**
  - Given input x: $F(x)$ easy to compute
  - Given output y: hard to find input x with $F(x) = y$
  - Basis for asymmetry of public-key algorithms

- **Trapdoor one-way function $F(x) = y$**
  - Given y and some secret: easy to find x with $F(x) = y$
  - Examples of secrets: prime factors, discrete logarithm
  - Basis for private key and decryption

Public Key Encryption

<u>Def</u>: a public-key encryption system is a triple of algs.(G, E, D)

- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$(pk, sk) output by G :

$$\forall m \in M: \quad D(sk, E(pk, m)) = m$$

Trapdoor functions (TDF)

Def: a trapdoor func. $X \longrightarrow Y$ is a triplet of efficient algs. (G, F, F^{-1})

- $G()$: randomized alg. outputs a key pair (pk, sk)
- $F(pk,\cdot)$: det. alg. that defines a function $X \longrightarrow Y$
- $F^{-1}(sk,\cdot)$: defines a function $Y \longrightarrow X$ that inverts $F(pk,\cdot)$

More precisely: $\forall (pk, sk)$ output by G

$$\forall x \in X: \quad F^{-1}(sk, F(pk, x)) = x$$

(G, F, F^{-1}) is secure if $F(pk, \cdot)$ is a "one-way" function:

can be evaluated, but cannot be inverted without sk

Review: arithmetic mod composites

Let $N = p \cdot q$ where p,q are prime

$Z_N = \{0,1,2,\ldots,N-1\}$; $(Z_N)^* = $ {invertible elements in Z_N}

Facts: $x \in Z_N$ is invertible $\Leftrightarrow$ $\gcd(x,N) = 1$

- Number of elements in $(Z_N)^*$ is $\varphi(N) = (p-1)(q-1) = N-p-q+1$

Euler's thm: $\boxed{\forall\ x \in (Z_N)^* \ : \ x^{\varphi(N)} = 1}$

The RSA trapdoor permutation

First published: Scientific American, Aug. 1977.

Very widely used:

- SSL/TLS: certificates and key-exchange
- Secure e-mail and file systems

… many others

The RSA trapdoor permutation

1. - Choose random primes ***p, q*** (≈1024 bits*) and compute* ***N = p.q***
2. - Compute Euler function **φ(N) = (p-1)(q-1)**
3. - Choose random encryption key ***e*** *with* ***gcd (e, φ(N))*** ***= 1***
- Compute decryption key $\mathbf{d = e^{-1} \bmod \varphi(N)}$
  - s.t. $\mathbf{e \cdot d = 1 \pmod{\varphi(N)}}$
  -

output pk = (N, e) , sk = (N, d)

$\mathbf{F(\,pk, x\,)}$: $\mathbf{RSA(x) = x^e}$ (in Z_N) = y

---

$\boxed{\mathbf{F^{-1}(\,sk, y)} = y^d\,;}$ $y^d = \mathbf{RSA(x)^d} = x^{ed} = x^{k\varphi(N)+1} = (x^{\varphi(N)})^k \cdot \mathbf{x} = x$

The RSA Algorithm Example

- Choose p = 3 and q = 11
- Compute n = p * q = 3 * 11 = 33
- Compute φ(n) = (p - 1) * (q - 1) = 2 * 10 = 20
- Choose e such that 1 < e < φ(n). Let e = 7
- Compute a value for d such that (d * e) % φ(n) = 1. One solution is d = 3 [(3 * 7) % 20 = 1]
- Public key is (e, n) => (7, 33)
- Private key is (d, n) => (3, 33)
- The encryption of $m = 2$ is $c = 2^7 \% 33 = 29$
- The decryption of $c = 29$ is $m = 29^3 \% 33 = 2$

Security of RSA

- **Main attack vectors against RSA**
  - Decrypting ciphertext *c directly*: $c = m^e \bmod n$
  - → *Difficulty of computing roots in modular arithmetic*
  - Deriving private key d: $d = e^{-1} \bmod \varphi(n)$ ← $(p-1)\cdot(q-1)$
  - → *Difficulty of computing prime factors from n*

- **Security (difficulty) depends on size of prime numbers**
  - Factorization of numbers up to 768 bits feasible
  - Keys with 2048 and more bits deemed secure
    - *(that is, ~600 decimal digits)*

Textbook RSA is insecure

Textbook RSA encryption:

- public key: **(N,e)**
- secret key: **(N,d)**

Encrypt: $\mathbf{c} \longleftarrow \mathbf{m^e}$ (in Z_N)

Decrypt: $\mathbf{c^d} \longrightarrow \mathbf{m}$

Insecure cryptosystem !!

- Is not semantically secure and many attacks exist

$\Rightarrow$ The RSA trapdoor permutation is not an encryption scheme !

Public-key encryption from TDFs

- (G, F, F^{-1}): secure TDF $X \longrightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H: X \longrightarrow K$ a hash function

We construct a pub-key enc. system (G, E, D):

Key generation G: same as G for TDF

Public-key encryption from TDFs

- (G, F, F^{-1}): secure TDF $X \longrightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H: X \longrightarrow K$ a hash function

<u>E(pk, m)</u> :

$x \xleftarrow{R} X, \quad y \leftarrow F(pk, x)$

$k \leftarrow H(x), \quad c \leftarrow E_s(k, m)$

output (y, c)

<u>D(sk, (y,c))</u> :

$x \leftarrow F^{-1}(sk, y),$

$k \leftarrow H(x), \quad m \leftarrow D_s(k, c)$

output m

In pictures:

$$\underbrace{F(pk,\ x)}_{\text{header}}\ \underbrace{E_s(\ H(x),\ \ m\ )}_{\text{body}}$$

<u>Security Theorem</u>:

If **(G, F, F^{-1})** is a secure TDF, **(E_s, D_s)** provides auth. enc.

and **H:** $X \longrightarrow K$ is a "random oracle"

then **(G,E,D)** is CCA^{ro} secure.

Incorrect use of a Trapdoor Function (TDF)

Never encrypt by applying F directly to plaintext:

E(pk, m) :

output $c \longleftarrow F(pk, m)$

D(sk, c) :

output $F^{-1}(sk, c)$

Problems:

- Deterministic: cannot be semantically secure !!
- Many attacks exist

Review: RSA pub-key encryption (ISO std)

(E_s, D_s): symmetric enc. scheme providing auth. encryption.

H: $x \to K$ where K is key space of (E_s, D_s)

- **G**(): generate RSA params: $pk = (N,e)$, $sk = (N,d)$
- **E**(pk, m): (1) choose random x in Z_N

  (2) $y \leftarrow RSA(x) = x^e$, $k \leftarrow H(x)$

  (3) output $(y, E_s(k,m))$
- **D**(sk, (y, c)): output $D_s(H(RSA^{-1}(y)), c)$

Key lengths

Security of public key system should be comparable to security of symmetric cipher:

Cipher key-size	RSA Modulus size
80 bits	1024 bits
128 bits	3072 bits
256 bits (AES)	**15360** bits

Implementation attacks

Timing attack: [Kocher et al. 1997] , [BB'04]
The time it takes to compute c^d (mod N) can expose d

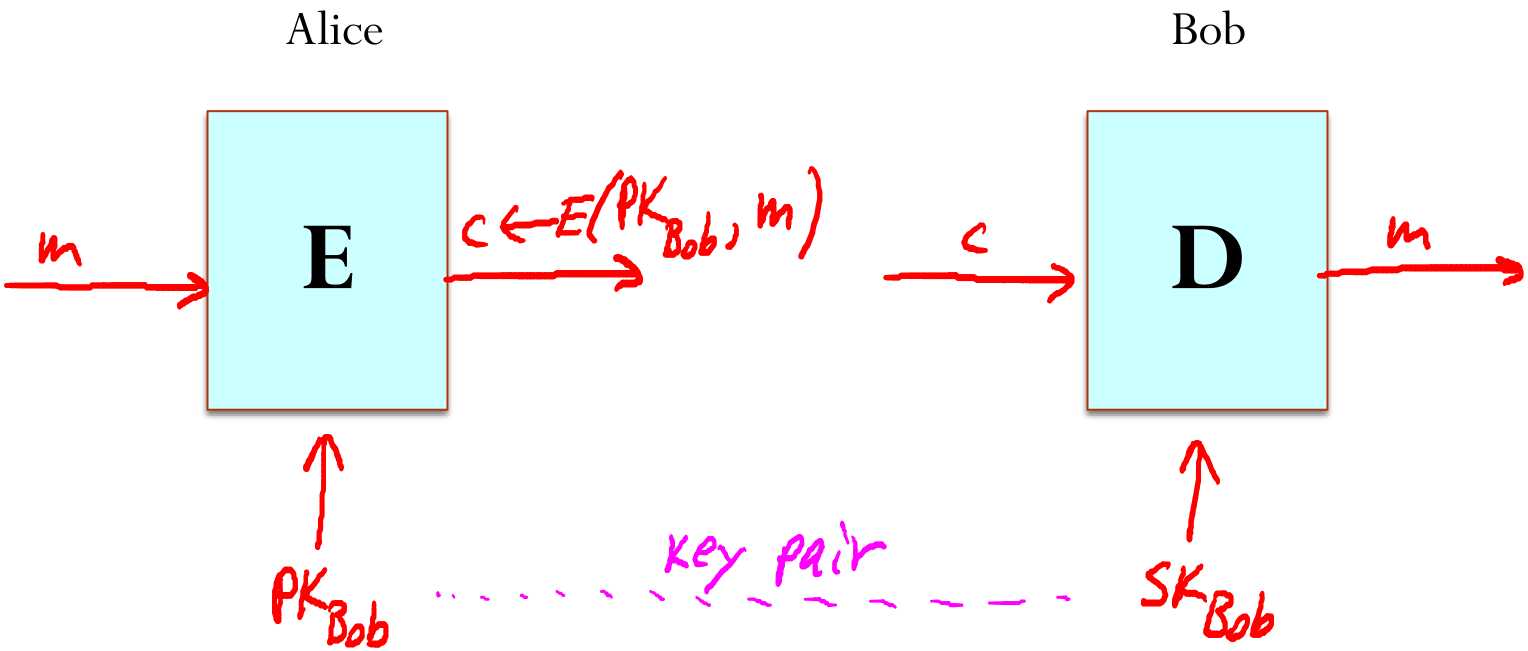
Power attack: [Kocher et al. 1999)
The power consumption of a smartcard while it is computing c^d (mod N) can expose d.

Faults attack: [BDL'97]
A computer error during c^d (mod N) can expose d.

A common defense:: check output. 10% slowdown.

Key Exchange with Public Key Encryption

Public key encryption



PK: public key, SK: secret key

Public key encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$(pk, sk) output by G :

$$\forall m \in M: \quad D(sk, E(pk, m)) = m$$

Establishing a shared secret

Alice **Bob**

(pk, sk) ⟵ G()

"Alice", pk

$\longrightarrow$

choose random
$x \in \{0,1\}^{128}$

"Bob", $c \leftarrow E(pk, x)$

$\longleftarrow$

$D(sk, c) \rightarrow x$

x: shared secret

Security (eavesdropping)

Adversary sees **pk, E(pk, x)** and wants **x** $\in M$

Semantic security $\Rightarrow$

adversary cannot distinguish

{ pk, E(pk, x), x } from { pk, E(pk, x), rand$\in$M }

$\Rightarrow$ can derive session key from x.

Note: protocol is vulnerable to man-in-the-middle

Insecure against man in the middle

As described, the protocol is insecure against **active** attacks

Alice | **MiTM** | **Bob**

$(pk, sk) \leftarrow G()$ $\qquad$ $(pk', sk') \leftarrow G()$

"Alice", pk $\longrightarrow$ | "Alice", pk' $\longrightarrow$

choose random
$x \in \{0,1\}^{128}$

"Bob", E(pk, x) $\longleftarrow$ | "Bob", E(pk', x) $\longleftarrow$

Public key encryption: constructions

Constructions generally rely on hard problems from number theory and algebra

Next module:

- Brief detour to catch up on the relevant background

Acknowledgements

Material in this lecture are taken from the slides prepared by:

- Prof. Dan Boneh (Standford)
- Prof. Dr. Konrad Rieck (Uni-Göttingen)