SCIENCE
PASSION
TECHNOLOGY

Modern Public Key Cryptography

Complexity Theory and Computational Hardness Assumptions

Lukas Helminger

March 3, 2020

Public-Key Cryptographic

Provable security idea:

- Breaking encryption (RSA, ECC, ...)
- as hard as solving hard problem (factoring, discrete logarithm)

Everything is an Algorithm

Encryption scheme:

- $key \leftarrow \text{KeyGeneration}(\cdot)$
- $c \leftarrow \text{Encryption}(m)$
- $m \leftarrow \text{Decryption}(c)$

Security Properties Adversary

Properties:

- Correctness

$$m = \text{Decryption}(\text{Encryption}(m)).$$

- $c =$Encryption(m) does not leak "any" information.
- unforgeability.

Adversary:

- runtime (poly-time).
- quantum

Hard problems

No crypto system relies on a proven hard problem.

Outline

Notation I

We denote

- $\mathbb{Z}$ as the set of integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$
- $\mathbb{N}$ as the set of natural numbers $\{0, 1, 2, \dots\}$
- $\mathbb{Z}_N$ as the set of integers modulo N
- $\mathbb{Z}_N^*$ as the set of invertible integers modulo N
- $\mathbb{P}$ as the set of prime numbers
- $(x_i)_{i=1}^n := (x_1, \dots, x_n)$

Notation II

We use $\mathbb{G}$ to denote a group.

- With $\mathbb{G} = \langle g \rangle$, we denote that g generates $\mathbb{G}$
- $|\mathbb{G}|$ denotes the order of a group
- κ... security parameter (in bits), e.g., RSA: $\kappa = 80$ bit $\approx$ 1024 bit modulus
- With $\mathcal{G}^\kappa = (\mathbb{G}, p, g)$, we denote the following setup:
  - p is a prime of bitlength κ, and
  - $\mathbb{G} = \langle g \rangle$ is a group with $|\mathbb{G}| = p$

Discrete Probability Distributions

Definition

A discrete probability distribution is a probability distribution that can take on a countable number of values.

Example: uniform distribution

$x \xleftarrow{R} X$ denotes x is drawn uniformly at random from X

Languages and Computational Problems

Definition	Example
Σ be a finite alphabet	$\{0, 1\}$
Σ^* is set of all strings of Σ	$\{0, 1, 10, 11, 01, \ldots\}$
A formal language L is a subset of Σ^*	strings of even length

- *Decision Problem:* Let $L \subseteq \Sigma^*$ be a language. On the input of $x \in \Sigma^*$, output `true` if $x \in L$ and `false` otherwise.
- *Search Problem:* Let $R \subseteq \Sigma^* \times \Sigma^*$ be a relation between inputs and outputs. On the input of $x \in \Sigma^*$, output $y \in \Sigma^*$ such that $(x, y) \in R$.

Oracle

Oracle

An oracle $\mathcal{O}$ is a black-box that can be used to solve a computational problem in one computational step.

Note: No analysis or modification of internal computations.

Let $\mathcal{A}$ be an algorithm (TM). We use $\mathcal{A}^{\mathcal{O}}$ to denote that $\mathcal{A}$ has access to oracle $\mathcal{O}$, e.g.

$$\overline{SAT} \in P^{SAT}.$$

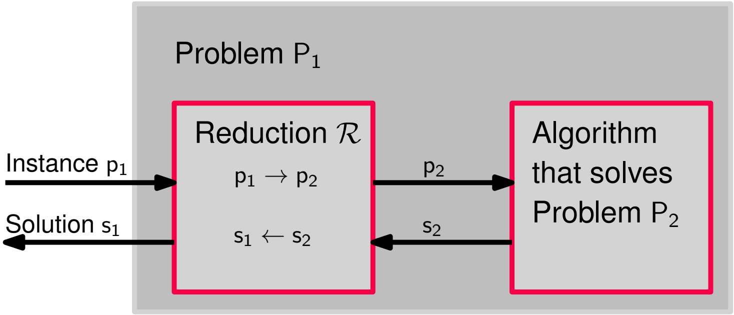
Probabilistic Polynomial Time (PPT)

A PPT algorithm $\mathcal{A}$ can make (polynomial many) random steps upon execution. The output of $\mathcal{A}$ is a random variable.

Find$(k, a_1, \dots, a_n)$:

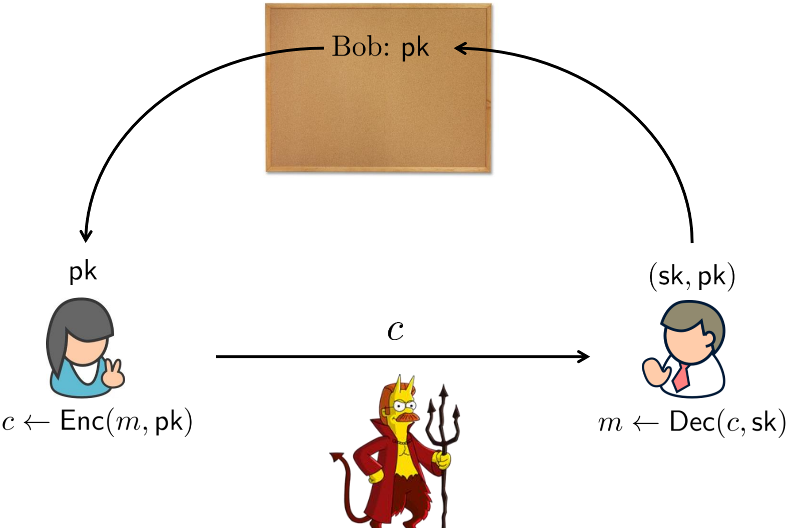
- Pick $i \in \{1, \dots, n\}$ randomly and set $x \leftarrow a_i$
- Scan $a_1, \dots, a_n$ and count the number m of a_j's s.t. $a_j \leq x$.
- If $m = k$ output x.
- If $m > k$ copy all elements a_j with $a_j \leq x$ in a new array L and run $\text{Find}_k(k, L)$
- If $m < k$ copy all elements a_j with $a_j > x$ in a new array L and run $\text{Find}_k(k - m, L)$

Reductions



We write $P_1 \leq P_2$, i.e., P_2 is at least as hard as P_1.

Algorithms in a Cryptographic Setting



Reductionist Security

Prove security by reduction to specific hard problem:

- Assume an PPT adversary $\mathcal{A}$ breaking a crypto system
- Show that there is an efficient reduction $\mathcal{R}$ from the crypto system to the hard problem

Goal: Crypto system is secure as long as factoring is hard.

Negligible Functions

Definition

A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is called negligible, if for every polynomial function $p : \mathbb{N} \to \mathbb{R}$, there is an $n_0 \in \mathbb{N}$ such that

$$\epsilon(n) \leq \frac{1}{p(n)} \qquad \forall\ n \geq n_0.$$

i.e. ϵ must be exponentially small $\forall\, n \geq n_0$.

Computational Hardness

Why: Information theoretically secure primitives are rare and often not very practical

Hard? Educated guess (heuristics).

Note

Assumptions can be analyzed independently of schemes.

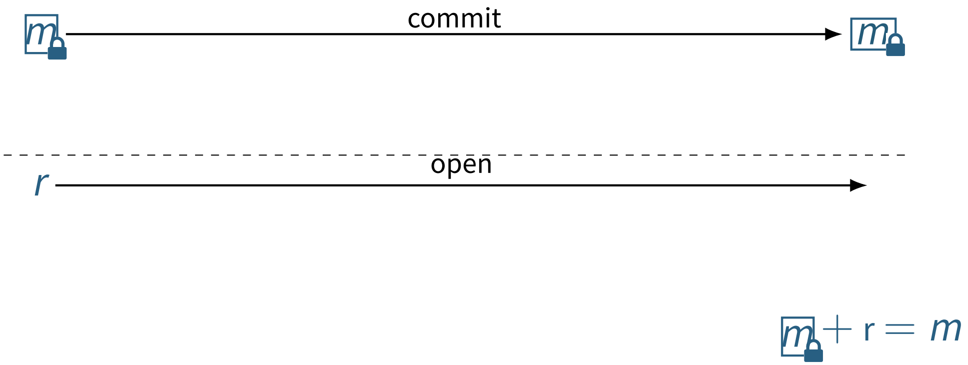
Discrete Logarithm Assumption

Let $\mathcal{G}^\kappa = (\mathbb{G}, p, g)$.

The discrete logarithm (*DL*) assumption states that forall PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that

$$\Pr\left[x \xleftarrow{R} \mathbb{Z}_p,\ x^* \leftarrow \mathcal{A}(\mathcal{G}^\kappa, g^x)\ :\ x = x^*\right] \leq \epsilon(\kappa).$$

Commitment Scheme



Hiding: Cannot learn m from Comm(m).

Binding: Cannot open $Comm(m)$ to two different messages.

Example: Commitment[1] under *DL*

Let $\mathcal{G}^\kappa = (\mathbb{G}, p, g)$ and additionally $\mathbb{G} = \langle h \rangle$.

Commitment C to message $m \in \mathbb{Z}_p$:

- Choose $r \xleftarrow{R} \mathbb{Z}_p^*$
- Compute $C \leftarrow g^m h^r$

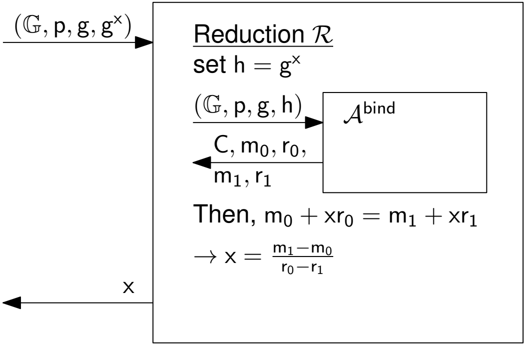
Binding: $\forall$ PPT $\mathcal{A}$ $\exists$ negl. $\epsilon(\cdot)$ such that

$$\Pr\left[(C, m_0, r_0, m_1, r_1) \leftarrow \mathcal{A}(\mathcal{G}^\kappa, h) \;:\; \begin{array}{r} C = g^{m_0} h^{r_0} \wedge \\ C = g^{m_1} h^{r_1} \wedge \\ m_0 \neq m_1 \end{array} \right] \leq \epsilon(\kappa).$$

[1] Pedersen Commitment

Example II

Prove binding by showing that an efficient adversary $\mathcal{A}^{\text{bind}}$ against binding can be used to construct an efficient adversary against *DL*.



Computational Diffie-Hellman Assumption

Let $\mathcal{G}^\kappa = (\mathbb{G}, p, g)$.

The computational Diffie-Hellman (*CDH*) assumption states that $\forall$ PPT $\mathcal{A}$ $\exists$ negl. $\epsilon(\cdot)$ such that

$$\Pr\left[x, y \xleftarrow{R} \mathbb{Z}_p,\ \ h \leftarrow \mathcal{A}(\mathcal{G}^\kappa, g^x, g^y)\ :\ h = g^{xy}\right] \leq \epsilon(\kappa).$$

Decisional Diffie-Hellman Assumption

Informally: Distinguish (g^x, g^y, g^{xy}) from (g^x, g^y, r), $r \in_R \mathbb{G}$.

Let $\mathcal{G}^\kappa = (\mathbb{G}, p, g)$. The decisional Diffie-Hellman (*DDH*) assumption states that $\forall$ PPT $\mathcal{A}$ $\exists$ negl. $\epsilon(\cdot)$ such that

$$\Pr\left[\begin{array}{r} x, y, z \xleftarrow{R} \mathbb{Z}_p,\ b \xleftarrow{R} \{0,1\}, \\ b^* \leftarrow \mathcal{A}(\mathcal{G}^\kappa, g^x, g^y, g^{(1-b)\cdot z + b\cdot xy})\ : \\ b = b^* \end{array}\right] \leq 1/2 + \epsilon(\kappa).$$

Relations between Assumptions

Theorem

Fix $\mathcal{G}^\kappa = (\mathbb{G}, p, g)$, then the following holds

$$DDH \leq_P CDH \leq_P DL.$$

Proof: Exercise.

Bilinear Maps I

Let $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2 = \langle g_2 \rangle$ and $\mathbb{G}_T$ be three groups of prime order p.

A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, with the following properties:

- Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a) \; \forall \, a, b \in \mathbb{Z}_p$
- Non-degeneracy: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, i.e., $e(g_1, g_2)$ generates $\mathbb{G}_T$

Digital Signature Scheme

KeyGen : Choose $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ where, $\mathbb{G}$, and $\mathbb{G}_T$ is a prime. Further, let g generate $\mathbb{G}$. Choose sk $\xleftarrow{R} \mathbb{Z}_p$ and pk $\leftarrow g^{\mathrm{sk}}$.

Sign(sk, m) : Output a signature $\sigma \leftarrow m^{\mathrm{sk}}$.

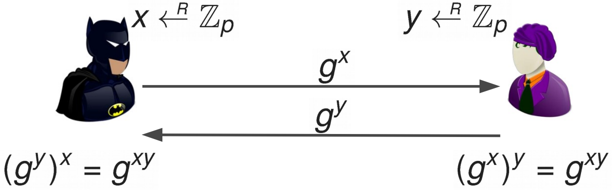
Verify(pk, m, σ) : Check if:

$$e(m, \mathrm{pk}) = e(\sigma, g).$$

Example - Bilinear Maps I

Recall: Diffie-Hellman key agreement

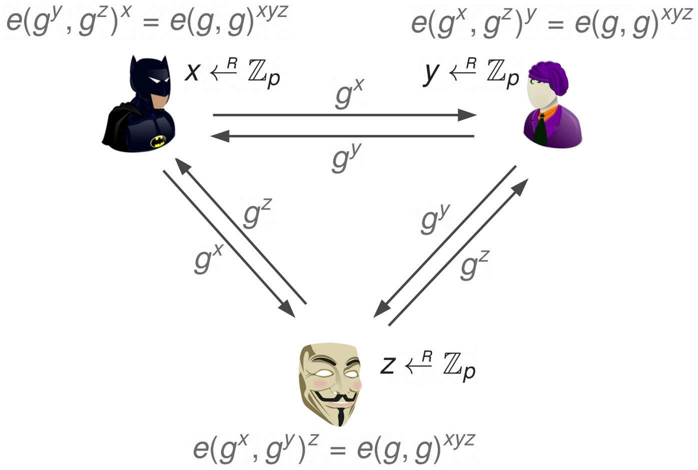
- $\mathcal{G}^{\kappa} = (\mathbb{G}, p, g)$



Example - Bilinear Maps II

Three party Diffie-Hellman key agreement

- $\mathcal{BG}^{\kappa} = (e, \mathbb{G}, \mathbb{G}_T, p, q)$



Bilinear Maps - Instantiations

Efficient instantiations using elliptic curve groups

- Here, $\mathbb{G}_1$ and $\mathbb{G}_2$ are prime order p elliptic curve subgroups
  - with point addition as group operation, and
  - $\mathbb{G}_T$ is the multiplicative order p subgroup of some extension field.
- Thus, often additive notation used for $\mathbb{G}_1$ and $\mathbb{G}_2$, e.g., for $P \in \mathbb{G}_1, P' \in \mathbb{G}_2, a, b \in \mathbb{Z}_p$ one would write

$$e(aP, bP') = e(P, P')^{ab} = e(bP, aP')$$

Bilinear Assumptions

Counterparts of *CDH*, *DDH* in the pairing setting.

Let $\mathcal{BG}_1^\kappa = (e, \mathbb{G}_1, \mathbb{G}_T, p, g_1)$. Then, $\forall$ PPT $\mathcal{A}$ $\exists$ negl. $\epsilon(\cdot)$ such that

- Computational bilinear Diffie-Hellman assumption (CBDH):

$$\Pr\left[x, y, z \xleftarrow{R} \mathbb{Z}_p, e(g_1, g_1)^{xyz} = \mathcal{A}(\mathcal{BG}_1^\kappa, g_1^x, g_1^y, g_1^z)\right] \leq \epsilon(\kappa).$$

- Decisional bilinear Diffie-Hellman assumption (DBDH)

$$\Pr\left[\begin{array}{l} x, y, z, w \xleftarrow{R} \mathbb{Z}_p, b \xleftarrow{R} \{0,1\}, \\ b^* \leftarrow \mathcal{A}(\mathcal{BG}_1^\kappa, g_1^x, g_1^y, g_1^z, \\ e(g_1, g_1)^{(1-b)\cdot w + b \cdot xyz}) : \\ \qquad\qquad b = b^* \end{array}\right] \leq 1/2 + \epsilon(\kappa).$$

Assumptions in Hidden-Order Groups

Let p, q be two appropriately chosen primes such that $N = pq$ is of bitlength κ. Then, $\forall$ PPT $\mathcal{A}$ $\exists$ negl. $\epsilon(\cdot)$ such that

- Integer factorization assumption:

$$\Pr\left[(p,q) \leftarrow \mathcal{A}(N) \;:\; N = p \cdot q\right] \leq \epsilon(\kappa)$$

- RSA assumption: Given e s.t. $\gcd(e, \varphi(N)) = 1$

$$\Pr\left[m \leftarrow \mathcal{A}(e,c,N) : m^e \equiv c \pmod{N}\right] \leq \epsilon(\kappa)$$

- Strong RSA assumption (s-RSA):

$$\Pr\left[(m,e) \leftarrow \mathcal{A}(c,N) : m^e \equiv c \pmod{N}\right] \leq \epsilon(\kappa)$$

Relations of Hidden-Order Assumptions

- It is easy to see that if one can factor, both RSA and s-RSA do not hold.
- Open problem: Show whether (s-)RSA is equivalent to factoring.

What you should know...

- Basic mathematical constructions: groups, generator, probability distribution
- Basic complexity theory: language, oracle, PPT
- High-level idea of reduction
- Discrete logarithm assumption
- Bilinear maps