CSC3631 Cryptography
- Intractability

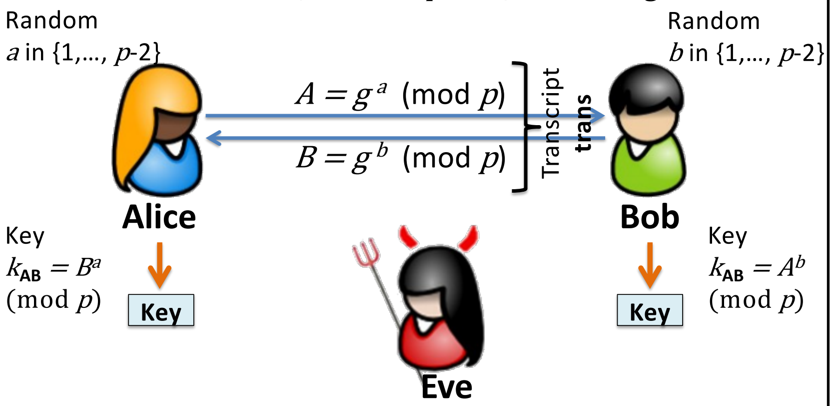
Thomas Gross

Foundational Laws

Fermat	Euler	e^{th} Root
$x^{p-1} = 1$ (mod p)	$x^{\varphi(N)} = 1$ (mod N)	$x^{1/e}$? (mod p)
Z_p	Z_N	Z_p
Gen Primes	RSA Crypto	RSA Crypto

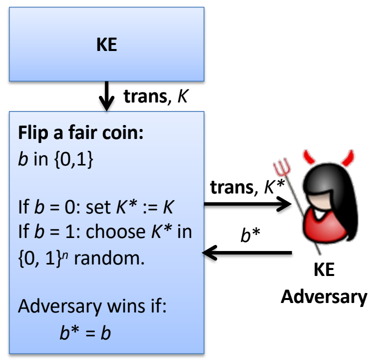
Diffie-Hellman in $(\mathbf{Z}_p)^*$

Given: large prime p; **generator** g

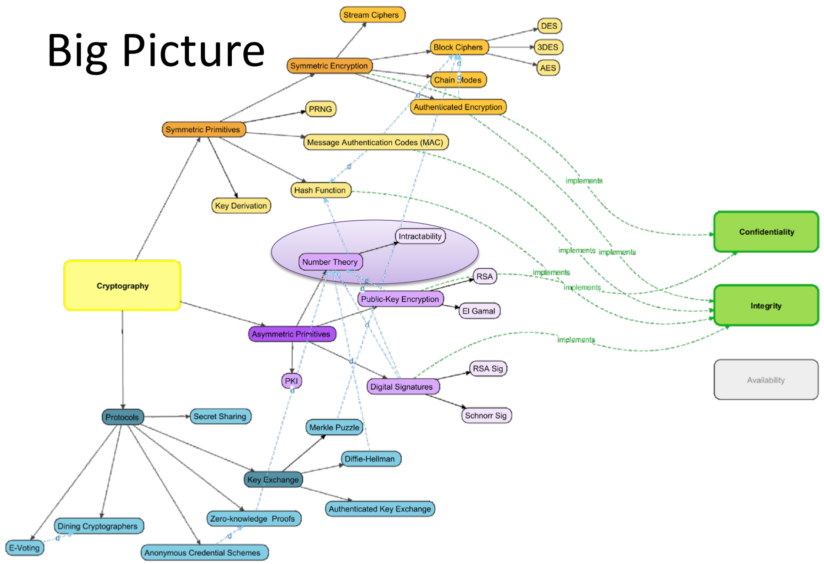


[Note: Full DH setup is in a sub-group G of $(\mathbf{Z}_p)^*$ with prime order q]

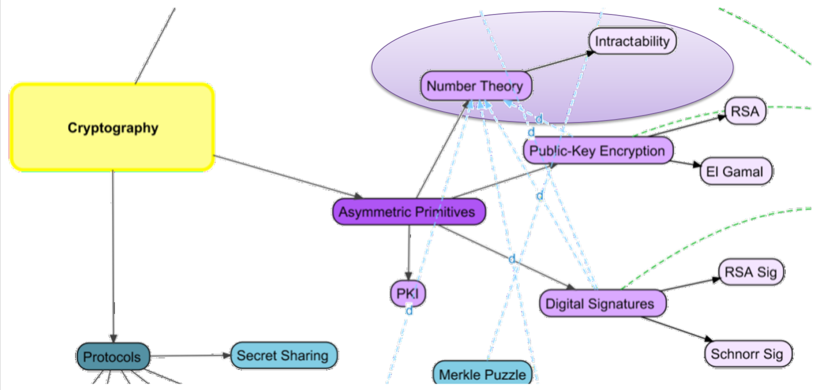
Key Exchange Security Definition



Big Picture



Big Picture



Roadmap

- Computing in $\mathbf{Z}_N$ (Easy Problems)
- Intractability
  - Factoring Problem
  - RSA Problem
  - Discrete Logarithm and Diffie-Hellman Problem

Goal for today:

- **How much work do computations take?**
- **What are hard problems in asymmetric crypto?**

Basic Operations

Let a and b be two **n-bit** integer. n = **len**(a)

Addition: a+b **O(n)**

Multiplication: a · b **O(n^2)**
Karatsuba **O(n$^{1.585}$)**

Division w/ remainder (a/b): a = b · q + r **O(n^2)**

Compute over the integers and do a modular reduction.

Naïve Exponentiation

Repeated multiplication: n = **len**(a) and **len**(x)

$$a^x = \underbrace{a \cdot a \cdot a \cdot \ldots a}_{x \text{ times}} \quad (\textbf{mod } N)$$

Number of multiplications :

Size of the intermediary value:

Example: RSA-Exponentiation (a, x and N: 3072 bit)

Size of the intermediary value

Square-and-Multiply

How to make modular exponentiation feasible?

Idea: Repeated squaring with modular reduction.

$$a^x \quad (\textbf{mod } N) \quad = \begin{cases} (a^{x/2})^2 (\textbf{mod } N) & \textbf{if } x \text{ even} \\ a \cdot (a^{(x-1)/2})^2 \ (\textbf{mod } N) & \textbf{if } x \text{ odd} \end{cases}$$

log x squarings and at most **log** x multiplications

Single multiplication $\mathbf{O}(n^2)$ ➔ ModExp: $\mathbf{O}(n^3)$

Example: Square-and-Multiply

$$a^x \quad (\textbf{mod } N) \ = \begin{cases} (a^{x/2})^2 \ (\textbf{mod } N) & \textbf{if } x \text{ even} \\ a \cdot (a^{(x-1)/2})^2 \ (\textbf{mod } N) & \textbf{if } x \text{ odd} \end{cases}$$

Compute:

$$\begin{aligned} 5^{17} &= 5^{[1\,0\,0\,0\,1]_{\text{bin}}} \\ &= \mathbf{5} \cdot (5^{[1\,0\,0\,0]_{\text{bin}}})^{\mathbf{2}} \\ &= 5 \cdot ((5^{[1\,0\,0]_{\text{bin}}})^2)^{\mathbf{2}} \\ &= 5 \cdot (((5^{[1\,0]_{\text{bin}}})^2)^2)^{\mathbf{2}} \\ &= 5 \cdot (((5)^2)^2)^{\mathbf{2}} \\ &= 762939453125 \end{aligned}$$

Efficient Computations

- Modular reduction: a (**mod** N)
- Add, subtract, multiply: $a+b$ (**mod** N)
  $a \cdot b$ (**mod** N)
- Check whether invertible: a^{-1}? (**mod** N)
- Compute inverse: a^{-1} (**mod** N)
- Exponentiation: a^b (**mod** N)

Polynomial algorithms known for all these operations.

Roadmap

- Computing in $\mathbf{Z}_N$ (Easy Problems)
- **Intractability**
  - Factoring Problem
  - RSA Problem
  - Discrete Logarithm and Diffie-Hellman Problem

Goal for today:

- **How much work do computations take?**
- **What are hard problems in asymmetric crypto?**

Intractability

What makes asymmetric cryptography secure?

Axiom 1:

All players are computationally limited.

(Only polynomial many steps)

Axiom 2:

Exist hard problems (not polynomial time).

Multiply (easy)

Example: **71, 89** **6319**

Factoring (hard)

When is a Problem Intractable?

What's the criterion for hard problems?

Play a game with Adversary **A**.

Setup

Inputs to Adversary **A**

Outputs from **A**

Success criterion for **A**

A problem is **hard** if there exists a game setup, such that all probabilistic and **polynomial-time** adversaries **A** only have **negligible success probability**.

Factoring

One of the most well studied hard problems

Given $N = p \cdot q$, $n = \mathbf{len}(N)$

where p and q are n-bit distinct primes.

Easy to compute N: $\mathbf{O}(n^2)$

Factoring: Find p and q, e.g., by trial division.

Hard. Most known methods: exponential in $n=\mathbf{len}(N)$

(Best known: Number Field Sieve $\mathbf{O}(N^{1/3}) = \mathbf{O}(2^{n^{1/3}})$)

Current Record in Factoring Challenge

RSA-768: factored by Kleinjung et al., 2009 (232 digits)

RSA-768 = 1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845251925573263034537315482685079170261221429134616704292143116022212404792747377940806653514195 974598569021434 13

p = 33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489 ×

q = 36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

RSA-1014 will become insecure within the decade.

[Source Wikipedia]

How hard is Factoring?

According to RSA Security:

Symmetric Cipher Key Size	Modulus Size
80 bits	**1024 bits**
128 bits	**3072 bits**

Best known algorithm to break factoring (and by that DH/RSA):

General Number Field Sieve

Expected running time **O**(**exp**(**log**(N)$^{1/3}$)

"Are 1024-bit RSA keys are dead?" Arien Lenstra: "The answer to that question is an unqualified yes."

[See Shoup2008, Section 15.5]

The Factoring Assumption

Setup: (N, p, q) ← **GenModulus**(1^n)

Input for Adversary **A**: N

Output of Adversary **A**: p' and q'

Adversary A success: if $p' \cdot q' = N$

Factoring is **hard** relative to GenModulus if all probabilistic and polynomial-time adversaries **A** only have negligible success probability.

Roadmap

- Computing in $\mathbf{Z}_N$ (Easy Problems)
- Intractability
  - Factoring Problem
  - **RSA Problem**
  - Discrete Logarithm and Diffie-Hellman Problem

Goal for today:

- **How much work do computations take?**
- **What are hard problems in asymmetric crypto?**

A more Practical Approach

Factoring is not an easy to use basis for crypto.

What other hard functions can we ask the adversary to compute?

Compute the e^{th} root in $(Z_N)^*$!

Example: What's $\mathbf{3^{1/5}}$ in $(\mathbf{Z_{35}})^*$**?**

How hard is computing the e^{th} Root?

Recall: How does that relate to computing the e^{th} root?

In $(Z_p)^*$: Easy if gcd$(e, p\text{-}1) = 1$

- **Choose** $d = e^{-1}$ **in** Z_{p-1} ➔ $d \cdot e = 1$ **in** Z_{p-1}
- $(c^d)^e = c^{d \cdot e} = c^{k(p-1)+1} = [c^{p-1}]^k \cdot c = c$

In $(Z_N)^*$: Hard

Believed to require factorization of N

The RSA Assumption

What's the basis of the RSA crypto system?

Setup: $(N, e, d) \leftarrow$ **GenRSA**(1^n), where $e \cdot d = 1$ **mod** $\varphi(N)$

Choose y from $(\mathbf{Z}_N)^*$

Input for Adversary **A**: N, e, y

Output of Adversary **A**: x in $(\mathbf{Z}_N)^*$

Adversary A success: if $x^e = y$ (**mod** N)

The RSA problem is **hard** relative to GenRSA if all probabilistic and polynomial-time adversaries **A** only have negligible success probability.

Roadmap

- Computing in $\mathbf{Z}_N$ (Easy Problems)
- Intractability
  - Factoring Problem
  - RSA Problem
  - **Discrete Logarithm and Diffie-Hellman Problem**

Goal for today:

- **How much work do computations take?**
- **What are hard problems in asymmetric crypto?**

What is the Discrete Logarithm?

Given an element h in $(\mathbf{Z}_p)^*$ with generator g,
find the x such that

$$g^x = h \quad (\textbf{mod } p)$$

Example: $(\mathbf{Z}_{17})^*$, g=3
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}

The Discrete Logarithm Assumption*

What's the basis of the DH and El Gamal crypto systems?

Setup: $((\mathbf{Z}_p)^*, q, g) \leftarrow$ **GenGroup**(1^n), where q = **ord**(g)
Choose h from $(\mathbf{Z}_p)^*$ by $h = g^{x'}$ (**mod** p)

Input for Adversary **A**: $(\mathbf{Z}_p)^*, q, g, h$
Output of Adversary **A**: x in $\mathbf{Z}_q$

Adversary A success: if $g^x = h$ (**mod** N)

The Discrete Logarithm problem is **hard** relative to GenGroup if all probabilistic and polynomial-time adversaries **A** only have negligible success probability.

[*) The Discrete Logarithm Assumption holds in arbitrary cyclic groups or order q.]

Decisional Diffie-Hellman Assumption*

What's the basis of the Diffie-Hellman key exchange?

Setup: $((\mathbf{Z}_p)^*, q, g) \leftarrow$ **GenGroup**(1^n), where q = **ord**(g)
Compute $h_1 = g^x$ (**mod** p) and $h_2 = g^y$ (**mod** p)

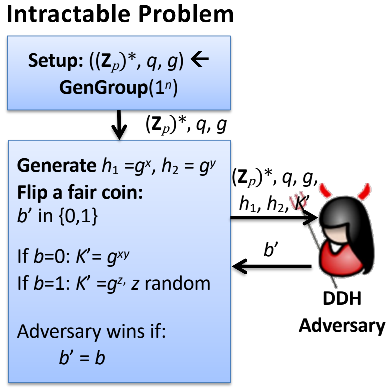
Input for Adversary **A**: $(\mathbf{Z}_p)^*, q, g, h_1, h_2, K$
where $K = g^z$ <u>or</u> $K = g^{xy}$
Output of Adversary **A**: Decision for g^z or g^{xy}

Adversary A success: if guessed type of K

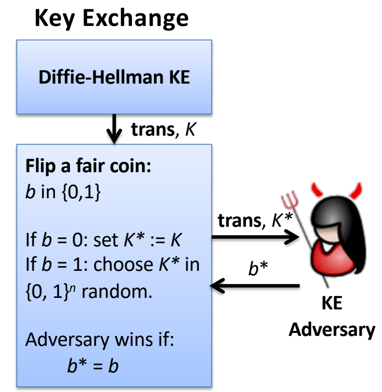
The Decisional Diffie Hellman problem is **hard** relative to GenGroup if all probabilistic and polynomial-time adversaries **A** only have negligible success probability to distinguish g^{xy} from a random number.

[*) In the key exchange lecture, we only considered the Computational Diffie-Hellman, as simplification.]

A Second View on the Decisional Diffie-Hellman Problem

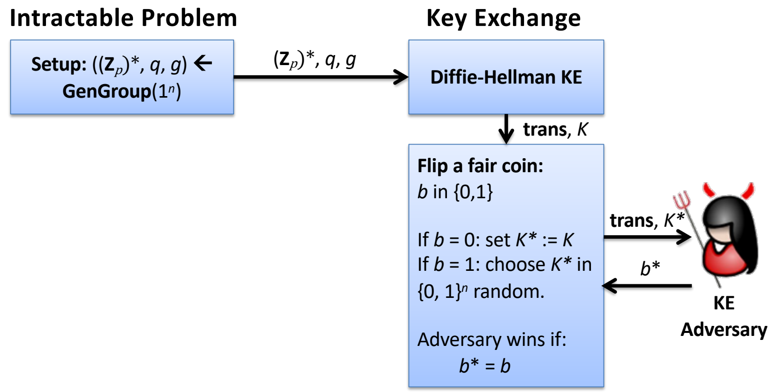


How to Prove Security of the Diffie-Hellman Key Exchange



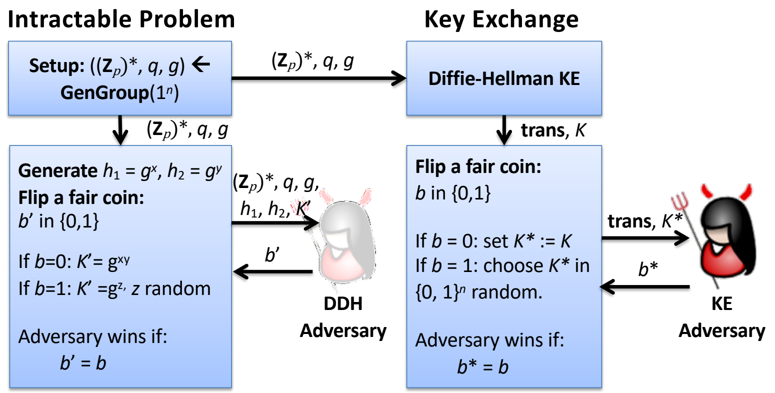
32

How to Prove Security of the Diffie-Hellman Key Exchange



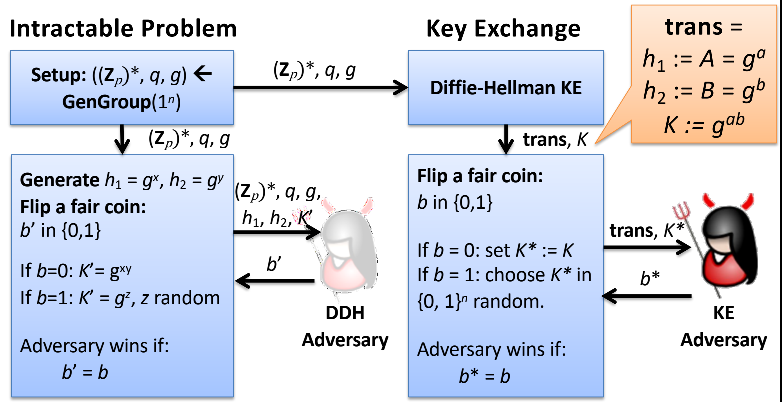
33

How to Prove Security of the Diffie-Hellman Key Exchange



34

How to Prove Security of the Diffie-Hellman Key Exchange



35

How to Prove Security of the Diffie-Hellman Key Exchange

Intractable Problem

Key Exchange

Setup: $((\mathbb{Z}_p)^*, q, g) \leftarrow$ **GenGroup**(1^n)

$(\mathbb{Z}_p)^*, q, g$

Generate $h_1 = g^x$, $h_2 = g^y$
Flip a fair coin:
b' in {0,1}

If b=0: $K' = g^{xy}$
If b=1: $K' = g^z$, z random

Adversary wins if:
$b' = b$

$(\mathbb{Z}_p)^*, q, g, h_1, h_2\ K'$

b'

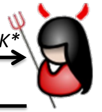
Reduction Translator

Set trans := h_1, h_2
Set $K^* := K'$

Set $b' = b^*$

trans, K^*

b^*



KE Adversary

36

How to Prove Security of the Diffie-Hellman Key Exchange

Intractable Problem

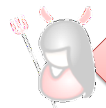
Setup: $((\mathbb{Z}_p)^*, q, g) \leftarrow$ **GenGroup**(1^n)

$(\mathbb{Z}_p)^*, q, g$

Generate $h_1 = g^x$, $h_2 = g^y$
Flip a fair coin:
b' in {0,1}

If b=0: $K' = g^{xy}$
If b=1: $K' = g^z$, z random

Adversary wins if:
$b' = b$



DDH Adversary

From a successful KE Adversary, we can efficiently construct a DDH Adversary

KE Adversary

37

Overview

Easy Land (Polynomial-Time)

Addition: $a+b$ (mod N) **Multiplication:** $a \cdot b$ (mod N)
Inverse: a^{-1} (mod N) **Exponentiation:** a^b (mod N)

Hard Land (Intractability)

Factoring	RSA	Discrete Log	Diffie-Hellman
What are p, q st. $p \cdot q = N$?	What's x st. $x^e = y$ (mod N)?	What's x st. $g^x = h$ (mod p)	Distinguish g^{xy} from g^z
Integers	**RSA Group** $(\mathbb{Z}_N)^*$, $N = pq$	**Subgroups of** $(\mathbb{Z}_p)^*$	**Subgroups of** $(\mathbb{Z}_p)^*$
Easy: Multiplication $N = p \cdot q$	**Easy: Exponentiation** $x^e = y$ (mod N)	**Easy: Exponentiation** $g^x = h$ (mod p)	**Easy: Exponentiation** $K = g^{xy}$ (mod p)
Hard: Find factors of N	**Hard:** Find the e^{th} root x given y and N.	**Hard:** Find the discrete logarithm x given h.	**Hard:** Decide whether K is DH or random

38