Cryptography
Winter term 2015/2016

Michael Nüsken

Bonn-Aachen International Center for Information Technology

2 November 2015 – 10 March 2016

Global Overview

Organizational
Introduction
Perfectly Secret Encryption

Symmetric-Key Cryptography

Symmetric-Key Encryption and Pseudorandomness, I
Practical Constructions of Block Ciphers
Symmetric-Key Encryption and Pseudorandomness, II
MACs and Collision-Resistant Hash Functions

Public-Key Cryptography

Symmetric-Key Management and Public-Key Revolution
Public-Key Encryption I
Number Theory
Factoring and Computing Discrete Logarithms
Public-Key Encryption, II
*Additional Public-Key Encryption Schemes
Digital Signature Schemes
*Public-Key Cryptosystems in the Random Oracle Model

Section 1 Overview

Organizational

- Webpage & mailing list
- Time & place
- Hand-in & exam

Introduction

Perfectly Secret Encryption

Organizational:

Webpage & mailing list

Course page

`https://cosec.bit.uni-bonn.de/students/teaching/15ws/15ws-crypto/`

Mailing list for discussions

`15ws-crypto@lists.bit.uni-bonn.de`
Subscribe today!

Organizational:

Time & place

Lectures

- Monday, 12^{45}-14^{15} **sharp**, b-it bitmax.
- Thursday, 12^{15}-13^{45} **sharp**, b-it bitmax.

Tutorial

- Monday, 14^{30}-16^{00} **sharp**, b-it bitmax.

Organizational:

Hand-in & exam

Hand-ins

- Out: Typically, Monday, 18^{00}.
- In: Friday, 23^{59}.

Bonus

- $\geq 50\%$: Admitted to the exam.
- $\geq 70\%$: One third bonus.
- $\geq 90\%$: Two third bonus.

Final exam

- 15 March 2016.
- $\geq 50\%$ of all points necessary to pass.
- If you pass, we apply the bonus.

Section 2 Overview

Organizational

Introduction

- Historical examples
  - Cesar's cipher
  - Shift cipher
  - Monoalphabetic substitution
  - The unbreakable cipher
  - Conclusions
- Kerckhoffs' principle
- Black-box view of encryption
- Basic principles of modern cryptography
- Attack scenarios

Perfectly Secret Encryption

Cesar's cipher

Replace each letter with its third successor: `a` becomes `D`, `b` becomes `E`, . . . Thus:

```
forest
IRUHVW
```

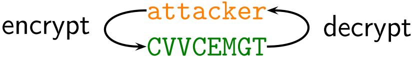
In modern language it's only a code.

Introduction:

Historical examples

Shift cipher

Replace each letter with its k-th successor.
For example with $k = 2$:



encrypt

attacker

CVVCEMGT

decrypt

But we only have 26 keys: $\{0, 1, 2, \dots, 25\} = \mathbb{N}_{<26}$.
Brute force[1] means: try all keys. That's done fast here.

[1] Brute force is no solution.

Introduction:

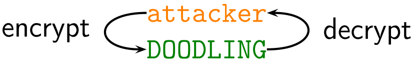
Historical examples

Monoalphabetic substitution

Instead of shifting the alphabet, we can permute it completely. Eg. we might choose the key:

```
abcdefghijklmnopqrstuvwxyz
DYLRNPHKSJIZEVUXFGAOMBCTQW
```

To encrypt or decrypt is easy:



Introduction:

Historical examples

Monoalphabetic substitution

Now we have $26!$ keys.[2] That's about $2^{88.4\top}$.

For comparison: one 4 Ghz CPU kernel runs $2^{56.8\bot}$ cycles per year or $2^{90.5\top}$ cycles since big bang[3]. So brute force would take

$$2^{31.6\top} \text{ years} = 0.23\bot \text{ ages of the universe}$$

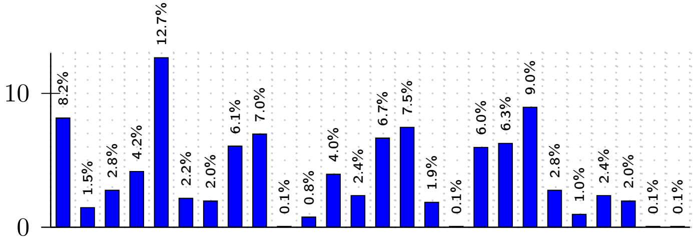
on a single such CPU kernel. Or one million such CPUs run for $3197.\bot$ years. So, brute force is out of reach.

[2] $26! = 403\,291\,461\,126\,605\,635\,584\,000\,000$.

[3] The age of the universe is $(13.799 \pm 0.021) \cdot 10^9$ years ($=2^{61.\top}$ s) acc.to. . .

Monoalphabetic substitution

Easy to break: frequency analysis.
In a typical English text the letters have the following frequencies:



This translates to frequencies of the ciphertext letters: The most frequent ciphertext letter corresponds most probably to the plaintext letter e. The second most... After a few steps the remaining letters follow by considering short words like the.

The unbreakable cipher

(Alberti ∼1467, Bellaso 1553, Vigenère ≈1850)

Aka. Vigenère cipher, polyalphabetic shift, . . .

Pick a word as key, say CRYPTO. Now, encrypt as follows:

```
useakeywordsaycryptotoencrypttheplaintext...
CRYPTOCRYPTOCRYPTOCRYPTOCRYPTOCRYPTOCRYPT...
WJCPDSANMGWGCPAGRDVFRDXBEIWEMHJVNATWPKCMM...
```

For each letter use the shift cipher according to the corresponding letter from the key, where A = 0, B = 1, . . . , Z = 25.

Already, for an alphabet of size 26 and key length of up to twenty letters there are $2^{94.1\Upsilon}$ keys.

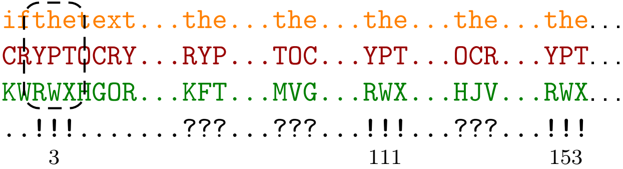
Still we can break it.

Introduction:

Historical examples

Kasiski attack (1863)

If the text is long enough, find repeated patterns of three, four or more letters. Consider the distances. Typically, these patterns are the encryption of the same plaintext pattern, like `the` or `of`.



```
ifthetext...the...the...the...the...the...
CRYPTOCRY...RYP...TOC...YPT...OCR...YPT...
KWRWXHGOR...KFT...MVG...RWX...HJV...RWX...
..!!!.......???...???...!!!...???...!!!
    3                       111       153
```

The distances are 108 and 42. Their greatest common divisor is 6. The key word `CRYPTO` has 6 letters. So that *is* the key length here.

Breaking knowing the key length

Once we know the key length κ, we split in groups of κ letters and then analyze the first letter of the groups, then the second and so on. These are generated by a shift cipher. So for example

```
KWRWXH GORXLZ QEETGC WXFUBB FICEXO VVBETH VVPCLC
HKFGXS HFSGHF OFPTES VKCGLQ QEQXWS TKFTWW UKYCVS
UKWEBQ CCJNMV GJCETH VVPCLO TVRWXS PTPNIH KFLDYH
JVQPFS RCYXGH GORETH VVPCEW MVRWXC TFD
```

The second letters are

```
WOEXIVVKFFKEKKKCJVVTFVCOVVF
```

The letter V has the largest frequency $\frac{7}{27} = 26.\bar{1}\%$. So it should be the e and thus the key letter is R.

Continuing we should eventually find the key word CRYPTO.

(Well, we find $\{$R CP$\}$ R $\{$LM YB$\}$ $\{$A Y CPST$\}$ T $\{$D O$\}$.)

Observation

Given the distribution p of letters in English we find that

$$\sum_{i\in\{\texttt{a},\ldots,\texttt{z}\}} p_i^2 \approx 0.065,$$

where p_i is the frequency of the letter i according to the distribution above.
For a random text however we would see

$$\sum_{i\in\{\texttt{a},\ldots,\texttt{z}\}} \left(\frac{1}{26}\right)^2 = 0.038\text{⫛}.$$

Better analysis of shift cipher

We can use that to find the best fitting key instead of 'only' looking at the most frequent letter(s).
Let q_i be the frequency of letter i in the ciphertext.
Slang: Consider the distribution q of the ciphertext.
Then we expect

$$I_k := \sum_{i \in \{\mathtt{A},\dots,\mathtt{Z}\}} p_i q_{i+k}$$

to be small for bad k and to be about 0.065 for the correct k.

Better analysis of the unbreakable cipher

Index of coincidence (Friedman 1922)

Keep in mind that given the distribution p of letters in English we find that

$$\sum_{i \in \{\mathtt{a},\ldots,\mathtt{z}\}} p_i^2 \approx 0.065,$$

where p_i is the frequency of the letter i according to the distribution above.
This is again true for the distribution q of a *shift cipher* encryption:

$$\sum_{i \in \{\mathtt{A},\ldots,\mathtt{Z}\}} q_i^2 \approx 0.065.$$

Just note that $q_{i+k} = p_i$ with the key k.

Better analysis of the unbreakable cipher

Consider the letters at positions 1, $1+\tau$, $1+2\tau$, $1+3\tau$ and so on.

If τ is a multiple of the key length κ, ie. $\kappa \mid \tau$, the distribution q of those letters should give

$$S_\tau = \sum_{i \in \{\mathtt{a},\ldots,\mathtt{z}\}} q_i^2 \approx 0.065.$$

If τ is *not* a multiple of the key length κ, ie. $\kappa \nmid \tau$, we should see a roughly uniform distribution with

$$S_\tau \approx \sum_{i \in \{\mathtt{a},\ldots,\mathtt{z}\}} \left(\frac{1}{26}\right)^2 = 0.038\perp.$$

Thus we can find the key length!

Introduction:

Historical examples

Summary

Cesar's cipher

Only a code (no key!).

Shift cipher

Only 26 keys.

Monoalphabetic substitution

$2^{88.4\Upsilon}$ keys $(= 26! = 403\,291\,461\,126\,605\,635\,584\,000\,000)$.
Still easy to break: frequency analysis.

The unbreakable cipher

We can break it. . . (Kasiski, Friedman)

Conclusions

- Ciphertext length needed for attacks depends on the size of the key space.
- Ciphertext-only vs. known-plaintext attacks: . . .
- Cipher design is tricky!

Introduction:

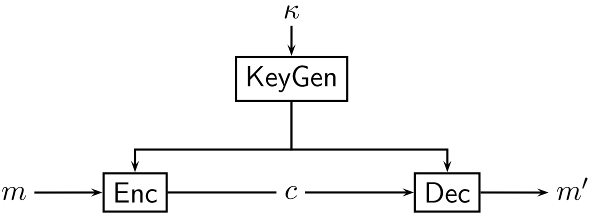
Kerckhoffs' principle

Kerckhoffs' principle

The attacker knows everything... but the key.

Introduction:

Black-box view of encryption



Correctness

For every security parameter κ and every message m we obtain $m' = m$.

Efficiency

Each box runs fast:

- at most a few seconds, say.
- polynomial time.

Security? Security! Security???

Principle 1: Exact definitions

We need rigorous, precise, exact definitions.

That is important

- . . . for design.
  Otherwise: how to know that we did it?
- . . . for usage.
  Otherwise: how to correctly use a system within a larger one?
- . . . for study.
  Otherwise: how to compare two systems?

Example: What is secure encryption?

~~Answer 1~~

~~An encryption scheme is secure if no attacker can find the *secret key* when given a ciphertext.~~

- Well, don't we want to protect the plaintext?
- Even worse: consider the scheme where KeyGen outputs a random κ-bit string and Enc and Dec merely output their inputs. Clearly, the attacker can never find the secret key even if he could obtain encryptions and decryptions as many as he desires. With this definition this system would be called secure. But it clearly is not![4]

[4] "But that's not what I meant!" Well: That's exactly the point...

Example: What is secure encryption?

~~Answer 2~~

~~An encryption scheme is secure if no attacker can find the *plaintext* when given a ciphertext.~~

- Better.
- But what if the scheme reveals 90% of the plaintext.
  Then it is not secure.

Example: What is secure encryption?

~~Answer 3~~

~~An encryption scheme is secure if no attacker can find *any character of the plaintext* when given a ciphertext.~~

- Looks good...
- But the scheme may still reveal whether your encrypted contract specifies a salary of less than $100\,000$ € or more. So this is still not enough.

Example: What is secure encryption?

~~Answer 4~~

~~An encryption scheme is secure if no attacker can derive *any meaningful information about the plaintext* when given a ciphertext.~~

- Looks even better…
- But: what is 'meaningful'?
  Well, we need to be more precise!

Example: What is secure encryption?

Answer 5

An encryption scheme is secure if no attacker can compute *any function of the plaintext* when given a ciphertext.

- That's best so far. . . .

Yet, it still does not specify everything. For example:

- Do we allow the attacker to obtain the decryption of other ciphertexts?
- And how many resources does the attacker have (time, memory, power, money)?

Principle 2:

Unproven assumptions must be precisely stated. And as ‘minimal’ as possible.

That is important because

- . . . almost all modern cryptographic schemes are only secure relative to some assumption.
- . . . only then we can validate or falsify them.
- . . . otherwise we cannot compare two schemes based on different assumptions.
- . . . only that allows security reductions.

Principle 3:

Cryptographic constructions must be accompanied by a precise security reduction.

That is important because

- proofs are better than intuition.

Relative security

Principle

We need rigorous, precise, exact definitions.

Principle

Unproven assumptions must be precisely stated. And as ‘minimal’ as possible.

Principle

Cryptographic constructions must be accompanied by a precise security reduction.

Introduction:

Attack scenarios

What can we say about the attacker?

Resources

The attacker's resources, ie. time, memory, power, money, must be bounded either

- polynomially wearing our asymptotic glasses, or
- by specifiable constants wearing our fixed size glasses.

Task

- Find the key. (UBK)
- Find the plaintext. (OW)
- Find some 'bit' of the plaintext. (semantic)
- Distinguish plaintexts. (IND)
- Modify a ciphertext. (NM)

Means

- Public-only attack (POA/KOA/COA).
- Known-plaintext attack (KPA).
- Chosen-plaintext attack (CPA).
- Chosen-ciphertext attack (CCA).

Section 3 Overview

Organizational

Introduction

Perfectly Secret Encryption

- The One-Time Pad (Vernam's cipher, 1917)
- Perfect secrecy

Perfectly Secret Encryption:

The One-Time Pad (Vernam's cipher, 1917)

Pick a random sequence as key as long as the plaintext. Now, encrypt as follows:

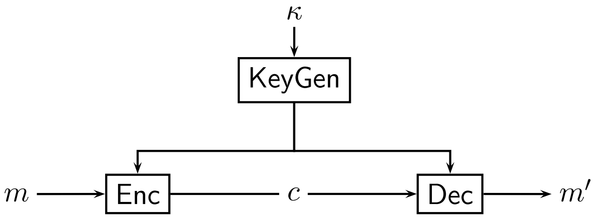
```
pickarandomsequenceaskeyaslongastheplaintext...
ZMGRSFGTLSFUWFBUVJIESVLLBYXDGMXTNYBNLXYRJLZJ...
OUIBSWGGOGRMAVVYILMEKFPJBQIRTSXLGFFCWXGECPWC...
```

For each letter use the shift cipher according to the corresponding letter from the key, where $\mathtt{A} = 0$, $\mathtt{B} = 1$, ..., $\mathtt{Z} = 25$.

- This we cannot break.
- No one can.
- And we can prove that.
- And it essentially is the only such cipher.

Perfectly Secret Encryption:

The One-Time Pad (Vernam's cipher, 1917)

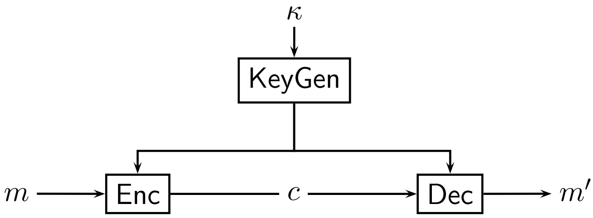


- KeyGen produces a random κ-bit string.
- $\text{Enc}(m, k) = m \oplus k$, bit-wise XOR of plaintext and key.
- $\text{Dec}(c, k) = c \oplus k$, bit-wise XOR of ciphertext and key.

And we have

- the key space $\mathcal{K} = \{0,1\}^\kappa$,
- the plaintext space $\mathcal{M} = \{0,1\}^\kappa$ and
- the ciphertext space $\mathcal{C} = \{0,1\}^\kappa$.

Candidate format



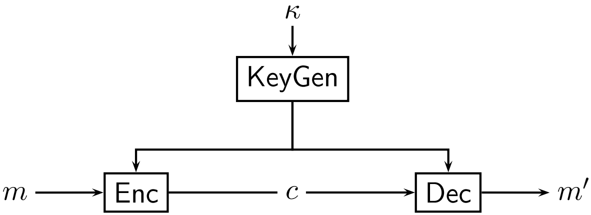
- KeyGen produces a random κ-bit string: $\mathcal{K} = \{0,1\}^{\kappa}$.
- Enc any algorithm, possibly probabilistic.
- Dec any algorithm, possibly probabilistic.

Correctness

$\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m.$

Perfectly Secret Encryption:

Perfect secrecy



κ

KeyGen

m

Enc

c

Dec

m'

Definition

An encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is *perfectly secret* if for every distribution over $\mathcal{M}$, every message $m \in \mathcal{M}$ and every ciphertext $c \in \mathcal{C}$ for which $\mathrm{prob}\left(C = c\right) > 0$ it holds that

$$\mathrm{prob}\left(M = m \,\middle|\, C = c\right) = \mathrm{prob}\left(M = m\right).$$

Perfectly Secret Encryption:

Perfect secrecy

Wlog. $\forall m \in \mathcal{M}\colon$ prob $(M = m) > 0$, $\forall c \in \mathcal{C}\colon$ prob $(C = c) > 0$.

Lemma

An encryption scheme (KeyGen, Enc, Dec) *is perfectly secret if and only if for every distribution over* $\mathcal{M}$, *every message* $m \in \mathcal{M}$ *and every ciphertext* $c \in \mathcal{C}$ *it holds that*

$$\text{prob}\left(C = c \,\middle|\, M = m\right) = \text{prob}\left(C = c\right).$$

Proof.

… □

Perfect indistinguishability

Lemma

An encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly secret iff for every distribution over* $\mathcal{M}$*, every* $m_0, m_1 \in \mathcal{M}$ *and every ciphertext* $c \in \mathcal{C}$ *it holds that*

$$\mathsf{prob}\left(C = c \,\middle|\, M = m_0\right) = \mathsf{prob}\left(C = c \,\middle|\, M = m_1\right).$$

Proof.

. . . □

Perfectly Secret Encryption:

Perfect secrecy

Indistinguishability game

- Prepare a key $k \leftarrow \mathsf{KeyGen}(\kappa)$ in $\mathcal{K}$.
- Choose a hidden bit $h \xleftarrow{🎲} \{0,1\}$ uniformly random.
- Prepare a *one-time* oracle $\mathcal{O}_{\mathsf{Test}}$ that when called with $m_0, m_1 \in \mathcal{M}$ the oracle returns $c \leftarrow \mathsf{Enc}_k(m_h)$.
- Call the attacker $\mathcal{A}$ with the oracle $\mathcal{O}_{\mathsf{Test}}$ and await a guess $h' \in \{0,1\}$.
- If $h = h'$ then ACCEPT else REJECT.

Theorem

An encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly secret iff for every attacker* $\mathcal{A}$ *we have* $\operatorname{prob}\left(\mathsf{Game}(\mathcal{A}) = ACCEPT\right) = \frac{1}{2}$.

Proof.

Perfectly Secret Encryption:

Perfect secrecy

Recall: for the One-Time Pad we have $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^\kappa$,
KeyGen picks an element if $\mathcal{K}$ uniformly at random,
$\mathsf{Enc}_k(m) = m \oplus k$, $\mathsf{Dec}_k(c) = c \oplus k$.

Theorem

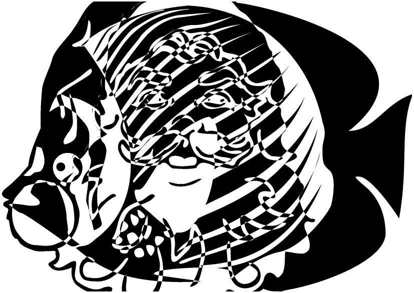
The one-time pad encryption scheme is perfectly secure.

Proof.

. . . □

Drawbacks

- Key must be uniformly random: expensive.
- Key can only be used once: $(m_0 \oplus k) \oplus (m_1 \oplus k) = m_0 \oplus m_1$.



- Key must be as long as the message.

Key must be as long as the message.

Lemma

If $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is perfectly secret then* $\#\mathcal{K} \geq \#\mathcal{M}$*.*

Proof.

… □

Unfortunately, we have no choice

The One-Time Pad is essentially the only perfectly secret one:

Theorem (Shannon's theorem, 1949)

Assume (KeyGen, Enc, Dec) *is an encryption scheme with* $\#\mathcal{K} = \#\mathcal{M} = \#\mathcal{C}$*. Then it is perfectly secret iff*

1. *The distribution of keys is uniform: Every key* $k \in \mathcal{K}$ *must be chosen with equal probability* $\frac{1}{\#\mathcal{K}}$ *by the algorithm* KeyGen*.*
2. *For every* $m \in \mathcal{M}$ *and* $c \in \mathcal{C}$ *there exists a unique key* $k \in \mathcal{K}$ *mapping* m *to* $c = \mathsf{Enc}_k(m)$*.*

Part I

Symmetric-Key Cryptography

Symmetric-Key Encryption and Pseudorandomness, I

Practical Constructions of Block Ciphers

Symmetric-Key Encryption and Pseudorandomness, II

MACs and Collision-Resistant Hash Functions

Section 4 Overview

Symmetric-Key Encryption and Pseudorandomness, I

- Computational Approach
- Defining Computationally-Secure Encryption (IND-POA)
- Pseudorandomness
- Constructing Secure Encryption Schemes

Practical Constructions of Block Ciphers

Symmetric-Key Encryption and Pseudorandomness, II

MACs and Collision-Resistant Hash Functions

Symmetric-Key Encryption and Pseudorandomness, I:

Computational Approach

- Perfect security essentially only with One-Time Pad.
- Necessarily, $\#\mathcal{K} \geq \#\mathcal{M}$.

⇒ Mathematically indecipherable, but impractical.

- Kerckhoffs: The system must be practically, if not mathematically, indecipherable.

⇒ RELAX!

Instead of perfect security where we consider attackers with arbitrary runtime and 100% success now:

1. Bound resources, ie. consider only efficient attackers.
2. Allow partial success, ie. consider also attackers that only 'win' with some non-negligible success probability.

Symmetric-Key Encryption and Pseudorandomness, I:

Computational Approach

Fixed-size glasses

Definition (Concrete approach)

Let $t, \varepsilon \in \mathbb{R}_{>0}$ be some constants.
A scheme is (t, ε)-secure iff every attacker running for time at most t succeeds with probability at most ε in breaking the scheme.

Examples

	t	ε
n-bit key	t	$t \cdot 2^{-n}$
128-bit key	2^{80}	2^{-48}
some attacker $\mathcal{A}$	4 years	ε
$\mathcal{A}$	8 years	¿2ε?
$\mathcal{A}$	2 years	¿$\frac{1}{2}\varepsilon$?

But: Which hardware? Moore's law?

Asymptotic glasses

Definition (Asymptotic approach)

A scheme is (asymptotically, polynomially) secure iff every attacker running in polynomial time succeeds with negligible probability in breaking the scheme.

- Polynomial time: $t(n) \in n^{\mathcal{O}(1)}$, ie. there exists a constant c there is an n_0 such that for $n \geq n_0$ we have $|t(n)| \leq n^c$.
- Negligible success: $\varepsilon(n)$ is eventually smaller than any inverse polynomial or $\varepsilon(n)^{-1}$ is eventually larger than any polynomial, ie. for any constant c and large n we have $|\varepsilon(n)| \leq n^{-c}$.
- Warning: Significant success: $\varepsilon(n)^{-1} \in n^{\mathcal{O}(1)}$.
  - You can never have negligible and significant.
  - But you can have non-negligible and non-significant.
  - Negligible implies non-significant.

Symmetric-Key Encryption and Pseudorandomness, I:

Computational Approach

Example for polynomial time with negligible success

Suppose we have a scheme that is secure and an attacker running n^3 minutes succeeds in breaking it with success probability $2^{40} \cdot 2^{-n}$.

- $n = 40$: Attacker runs $45.$Ψ days for success 1.
- $n = 50$: Attacker runs $87.$Ƭ days for success 2^{-10}.
- $n = 500$: Attacker runs $238.$Ƭ years for success 2^{-460}.

Examples for negligible and significant functions

- 2^{-n} is negligible.
- $2^{-\sqrt{n}}$ is negligible.
- $n^{-1\,000\,000\,000\,000}$ is not negigible.
- $n^{-\log_2 n}$ is negligible.
- $n^{-\log_2 \log_2 \log_2 \log_2 \log_2 n}$ is negligible.
- Let $f(n) = 2^{-n}$ for even n and $f(n) = 1$ otherwise. This f is not negligible and not significant.

Lemma

Let f_1, f_2 be negligible functions. Then

1. *The function f_3 with $f_3(n) = f_1(n) + f_2(n)$.*
2. *For any positive polynomial p, the function f_4 with $f_{4(n)} = p(n) \cdot f_1(n)$ is negligible.*

Necessity of relaxations

Practical systems must have $\mathcal{K}$ much smaller than $\mathcal{M}$.
Then two attacks are always possible:

- Brute force attack: …runtime $\#\mathcal{K}$, success 1.
- Guessing attack: …runtime 1, success $\frac{1}{\#\mathcal{K}}$.
  Side remark 'amplification': we may call this guessing attack repeatedly until we are successful. …runtime $\#\mathcal{K}$, success 1.

Consequences

- We must restrict attackers and their success.
- $\#\mathcal{K}$ must be 'larger' than the attackers runtime.

Efficient computation

We need a stable model that does not depend on the computer or the programming language or the mathematical computation model.

Church-Turing thesis: All intuitively good models are equivalent.

Our refererence are probabilistic polynomial-time interactive Turing-machines.

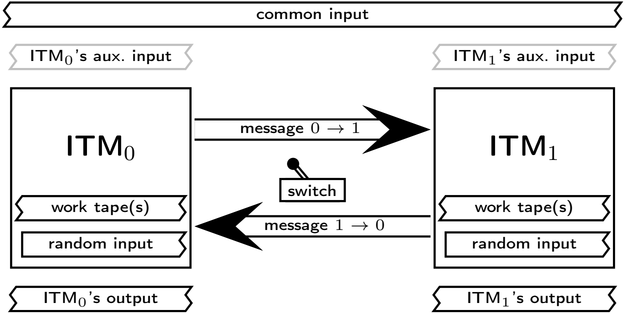


Figure: A linked pair of interactive TMs

Randomness

Why randomness? Well, everything else is predictable.

How to obtain randomness?

Theory: We just assume to have a tape with random bits.
Practice:

- Software random bit generators (entropy collectors).
  - `/dev/random`: $2^{5.6\perp}$ bit/sec.
- Hardware random bit generators (true randomness).
  - PRG310-4: up to $2^{18.\top}$ bit/sec.
- Pseudo-random bit generators.
  - RSA-based: 2^{24} bit/sec,
  - AES-based: $\gg 2^{30}$ bit/sec,
  - LFSR (not good for crypto): $\approx 2^{39}$ bit/sec.

Quality?

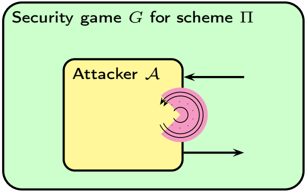
Reductions

- . . . prove security relative to some problem X.
- Reductions are unavoidable at present
  since we are still unable to prove that any of the *relevant* problems cannot be solved by a polynomial time algorithms.
- But we know: any such bound implies the existence of a one-way function, and

Theorem

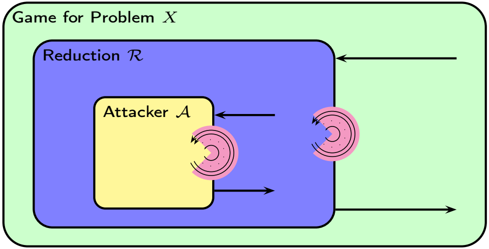
If one-way functions exist then $\mathcal{P} \neq \mathcal{NP}$.

Reductions



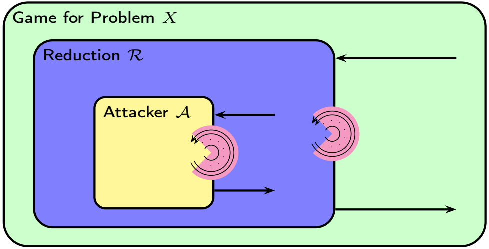
- Given an efficient attacker $\mathcal{A}$: runtime $t(n)$, success $\varepsilon(n)$.
    - It assumes to play a given security game G, which describes a break for some scheme Π.

Reductions



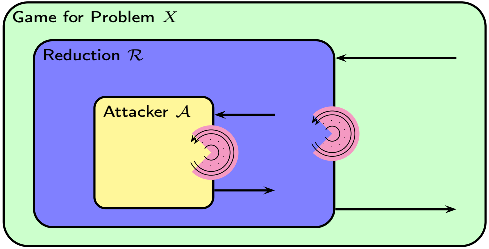
- We let it play against our reduction $\mathcal{R}$.
  - We must ensure that $\mathcal{A}$ cannot detect a difference.
  - We may manipulate input and oracles.
  - We may use the answer.
  - The reduction $\mathcal{R}$ tries to solve some problem X.

Reductions



- Assume: the reduction solve problem X with probability at least $\frac{1}{n^c}$ provided the attacker wins the original game.
- Runtime polynomial, success $\frac{\varepsilon(n)}{n^c}$.
- Thus: If $\mathcal{A}$ is successful, ie. $\varepsilon(n)$ is not negligible, then also $\mathcal{R}$ is successful, ie. $\frac{\varepsilon(n)}{n^c}$ is not negligible.

Reductions



Short

If $\mathcal{A}$ is successful then we can solve the problem X.

Theorem (Relative security)

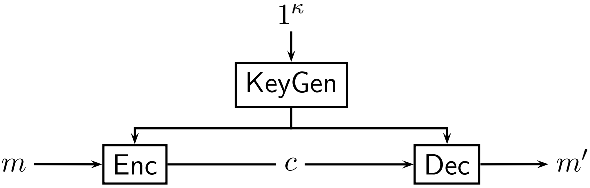
If the problem X is hard then the scheme is secure in the sense of the security game G.

Symmetric-Key Encryption and Pseudorandomness, I:

Defining Computationally-Secure Encryption (IND-POA)

Definition

A symmetric-key encryption scheme is a tuple $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$



such that

- Correctness: For every κ and $k = \mathsf{KeyGen}(1^\kappa)$ and for every message $m \in \mathcal{M}$ we have $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$.
- Efficiency: All algorithms (or protocols) run in polynomial time.
- Security: Well, . . .

Symmetric-Key Encryption and Pseudorandomness, I:

Defining Computationally-Secure Encryption (IND-POA)

Indistinguishability game G^{IND}

- Prepare a key $k \leftarrow \mathsf{KeyGen}(1^\kappa)$ in $\mathcal{K}$.
- Choose a hidden bit $h \overset{⚄}{\longleftarrow} \{0,1\}$ uniformly random.
- Prepare a *one-time* oracle $\mathcal{O}_{\mathsf{Test}}$ that when called with $m_0, m_1 \in \mathcal{M}$ the oracle returns $c \leftarrow \mathsf{Enc}_k(m_h)$.
- Call the attacker $\mathcal{A}$ with input 1^κ and the oracle $\mathcal{O}_{\mathsf{Test}}$. Await a guess $h' \in \{0,1\}$.
- If $h = h'$ then ACCEPT else REJECT.

IND-POA security

Definition

A symmetric-key encryption scheme Π is indistinguishable in the presence of an eavesdropper[5] iff for each probabilistic polynomial-time attacker $\mathcal{A}$ the advantage

$$\mathsf{adv}_{\Pi}^{\mathsf{IND}}(\mathcal{A}) = \left| \mathsf{prob}\left(G^{\mathsf{IND}}(\mathcal{A}) = \mathsf{ACCEPT} \right) - \frac{1}{2} \right|$$

is negligible.

[5] IND-POA = INDistinguishable under Public Only Attack

Alternative IND-POA security

Definition

A symmetric-key encryption scheme Π is indistinguishable in the presence of an eavesdropper[5] iff for each probabilistic polynomial-time attacker $\mathcal{A}$ the function

$$\left| \text{prob}\left(G^{\text{IND}}(\mathcal{A}) = \text{ACCEPT} \,\middle|\, h = 0\right) - \text{prob}\left(G^{\text{IND}}(\mathcal{A}) = \text{REJECT} \,\middle|\, h = 1\right) \right|$$

is negligible.

[5] IND-POA = INDistinguishable under Public Only Attack

Semantic security

Recall

Answer 5

An encryption scheme is secure iff no attacker can compute any function of the plaintext when given a ciphertext.

Why did we not use this formulation?

- It is difficult to handle. We have to consider any function.
- It turns out to be equivalent to the previous definition.

Semantic security

Theorem

If $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is IND-POA-secure then for each probabilistic polynomial-time attacker* $\mathcal{A}$ *and all* i *the advantage*

$$\left| \text{prob}\left(\begin{array}{l} \mathcal{A}(1^\kappa, \mathsf{Enc}_k(m)) \\ = \mathsf{bit}_i(m) \end{array} \right) - \frac{1}{2} \right|$$

is negligible.

Game $G^{\mathsf{bit}_i\text{-semantic}}$

- Prepare a key $k \leftarrow \mathsf{KeyGen}(1^\kappa)$ in $\mathcal{K}$ and a message $m \xleftarrow{🎲} \mathcal{M}$.
- Prepare a *one-time* oracle $\mathcal{O}_{\mathsf{Test}}$ that when called with no input the oracle and returns $c \leftarrow \mathsf{Enc}_k(m)$.
- Call the attacker $\mathcal{A}$ with input 1^κ and the oracle $\mathcal{O}_{\mathsf{Test}}$. Await a guess $b' \in \{0, 1\}$.
- If $\mathsf{bit}_i(m) = b'$ then ACCEPT else REJECT.

Symmetric-Key Encryption and Pseudorandomness, I:

Defining Computationally-Secure Encryption (IND-POA)

Semantic security

When trying to generalize this theorem to arbitrary functions f instead of bit$_i$ this gets tricky where picking m_0 and m_1.

Definition

A symmetric-key encryption scheme Π is
semantically secure in t
eavesdropper iff for ea
polynomial-time attac
probabilistic polynomi
that for each efficient
and all polynomial-tin
and h the advantage

$$\text{adv}_{\Pi}^{\text{semantic}}(\mathcal{A})$$
$$= \Big|\ \text{prob}\ \big(G^{\text{semantic}}(\mathcal{A}(h(m),\mathcal{O}_{\text{Test}})) = \text{ACCEPT}\big)$$
$$- \text{prob}\ \big(G^{\text{semantic}}(\mathcal{A}'(h(m))) = \text{ACCEPT}\big)\ \Big|$$

is negligible when m is chosen according to M.

Theorem

A symmetric-key encryption scheme is indistinguishable in the presence of an eavesdropper
iff
it is semantically secure in the presence of an eavesdropper.

Game G^{semantic}

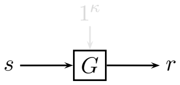
- Prepare a key $k \leftarrow \text{KeyGen}(1^\kappa)$ in
sage $m = M()$
I from $\mathcal{M}$.
-*time* oracle $\mathcal{O}_{\text{Test}}$
led with no input the
urns $c \leftarrow \text{Enc}_k(m)$.
ker $\mathcal{A}$ with input 1^κ,
oracle $\mathcal{O}_{\text{Test}}$. Await a
$1\}$.
then ACCEPT else
REJECT.

Why pseudorandomness first?

- Intuition: If ciphertext looks random, no attacker can learn from it.
- XOR with a pseudorandom string may be an alternative to the One-Time-Pad.

Pseudorandom generator

Definition



The expansion factor ℓ is a polynomial function and the fixed-length generator G is a deterministic polynomial-time algorithm that for input $s \in \{0,1\}^\kappa$ outputs a bitstring of length $\ell(\kappa)$.

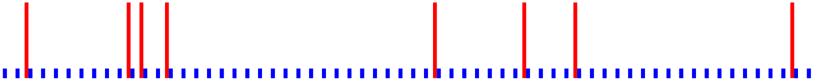
Now, G is a fixed-length pseudorandom generator iff

1. Expansion: $\ell(\kappa) > \kappa$.
2. Pseudorandomness: For each probabilistic polynomial-time distinguisher $\mathcal{D}$ the advantage
$$\mathrm{adv}_G(\mathcal{D}) = \left|\mathrm{prob}\left(\mathcal{D}(G(s)) = 1\right) - \mathrm{prob}\left(\mathcal{D}(r) = 1\right)\right|$$
is negligible. Here, $s \xleftarrow{\text{⚄}} \{0,1\}^\kappa$ and $r \xleftarrow{\text{⚄}} \{0,1\}^{\ell(\kappa)}$ are chosen uniformly at random.

Exercise

Formulate pseudorandomness with a game.

The output of a pseudorandom generator is far from random: Say, $\kappa = 3$, $\ell(\kappa) = 6$. Then the distributions of r and $G(s)$ may look as follows:



Thus with enough time a simple algorithm can detect the difference.

Brute force attack

Just consider the algorithm $\mathcal{D}_{\text{brute force}}$ that tests whether its input r equals $G(s)$ for some $s \in \{0,1\}^{\kappa}$. If so it answers 1, otherwise 0. That takes time 2^{κ} and has best possible advantage

$$\mathsf{adv}_{G}(\mathcal{D}_{\text{brute force}}) \geq 1 - 2^{\kappa-\ell(\kappa)} \geq \frac{1}{2}.$$

$\Rightarrow$ The seed must be long enough.

No efficient attack

However, no fast algorithm should be able to detect this difference. That's the definition of pseudorandomness.

Theorem

Pseudorandom generators exist $\iff$ *one-way function exist.*

Predictors (prophets) and postdictors (historians)

A predictor $\mathcal{P}$ predicts bit i of $G(s) \in \{0,1\}^{\ell(\kappa)}$ given bits $1..i-1$.

Theorem (Yao)

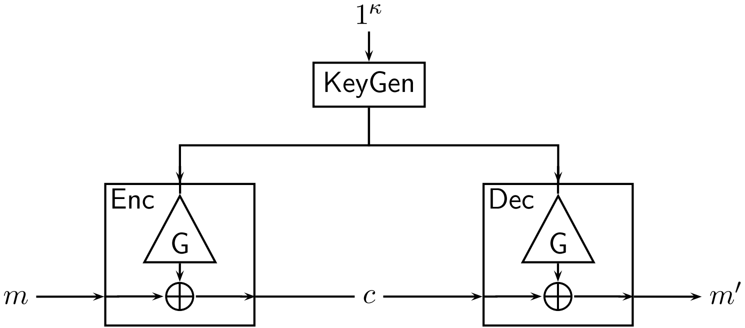
1. *If there is a predictor $\mathcal{P}$ for a generator G with advantage*

$$\begin{aligned}\mathsf{adv}_{G^{\mathsf{predict}}}(\mathcal{P}) = \Big|\, & \mathsf{prob}\,(\mathcal{P}(G(s)[1..(i-1)]) = G(s)[i]) \\ - \; & \mathsf{prob}\,(\mathcal{P}(r[1..(i-1)]) = r[i]) \qquad \Big|\end{aligned}$$

   then there is a distinguisher $\mathcal{D}$ with the same advantage.
2. *Given a distinguisher $\mathcal{D}$ there is a predictor $\mathcal{P}$ with advantage* $\mathsf{adv}_{G^{\mathsf{predict}}}(\mathcal{P}) \geq \frac{1}{\ell(\kappa)}\,\mathsf{adv}_G(\mathcal{D})$.

Reverse it: postdictors.

An encryption scheme Π_G from a generator G



An encryption scheme Π_G from a generator G

KeyGen

Input: 1^κ.
Output: $k \in \{0,1\}^\kappa$.

- Pick $k \xleftarrow{\text{⚅}} \in \{0,1\}^\kappa$.

Enc

Input: k, m.
Output: c.

- $c \leftarrow G(k) \oplus m$.

Dec

Input: k, c.
Output: m.

- $m \leftarrow G(k) \oplus c$.

Indistinguishability from Pseudorandomness

Theorem

If G is a pseudorandom generator then the just constructed fixed-length encryption scheme Π_G is indistinguishable in the presence of an eavesdropper.

Concrete security

Notice that the previous theorem and proof can be carried out with conrete bounds for time and advantage:

Theorem

If G is a (t, ε)-pseudorandom generator[6] *then Π_G is $(t - c, \varepsilon)$-indistinguishable*[7] *for some (small) constant c.*

[6] Ie. each distinguisher running in time t has advantage at most ε.

[7] Ie. each attacker running in time $t - c$ has advantage at most ε.

Section 5 Overview

Symmetric-Key Encryption and Pseudorandomness, I

Practical Constructions of Block Ciphers

- Substitution-Permutation Networks
- AES
- Feistel Networks
- DES
- Increasing the Key Length of a Block Cipher
- Brief look: differential and linear cryptanalysis
- Summary
- Modes of operation

Symmetric-Key Encryption and Pseudorandomness, II

MACs and Collision-Resistant Hash Functions

Practical Constructions of Block Ciphers:

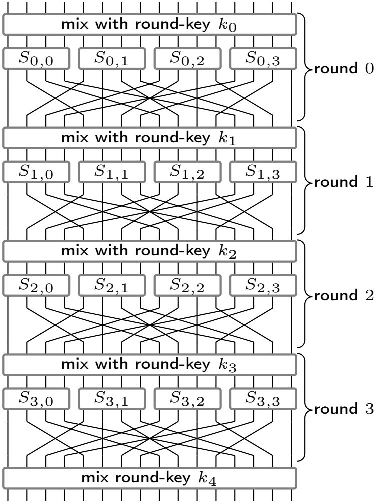
Substitution-Permutation Networks

Encryption is done by iterating rounds consisting of

- Mix (eg. XOR) with round key.
- Parallel S-box application.
- Permutation.

The S-box is the only non-linear component.

⇒ Confusion and diffusion.



Practical Constructions of Block Ciphers:

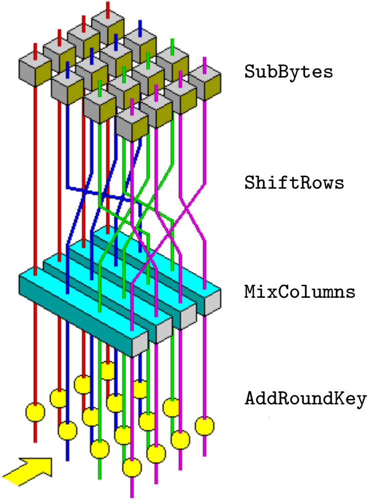
AES

AES

1997: NIST announces competition for Advanced Encryption Standard.
Submissions: 15.
Finalist: 5.

- Rijndael (SPN; Joan Daemen & Vincent Rijmen).
- Serpent (SPN).
- Twofish (Feistel).
- RC6 (Feistel).
- MARS (Feistel; IBM).

2000: Rijndael is selected as AES.
2002: AES effective.



©John Savard (1999)

AES

The field $\mathbb{F}_{2^8}$

$\mathbb{F}_{2^8} \ni a = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7$,
where $a_i \in \mathbb{F}_2 = \{0, 1\}$.
Representation: 8 bits for an element $= 1$ byte.
Addition: XOR, $(a + b)_i = a_i + b_i$.
Multiplication: as for polynomials modulo $x^8 + x^4 + x^3 + x + 1$.
Example 57 · 83 = `C1`**:**

$$\begin{aligned}(x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) &= x^{13} + x^{11} + x^9 + x^8 + x^7 + \\ &\quad x^7 + x^5 + x^3 + x^2 + x + \\ &\quad x^6 + x^4 + x^2 + x + 1 \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \\ &= x^7 + x^6 + 1 \mod x^8 + x^4 + x^3 + x + 1.\end{aligned}$$

Field: You can divide by every non-zero element.

AES

The S-box

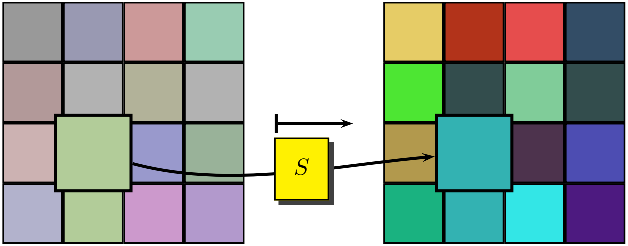
$$\begin{array}{cccccc}
 & \mathbb{F}_{2^8} & \longrightarrow & \mathbb{F}_{2^8} & \longrightarrow & \mathbb{F}_{2^8}, \\
\boxed{S}: & y & \longmapsto & y^{-1} \mathrel{\hat{=}} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} & \longmapsto & \begin{bmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}
\end{array}$$

Highly nonlinear:
$y \mapsto \mathtt{05} \cdot y^{254} + \mathtt{09} \cdot y^{253} + \mathtt{F9} \cdot y^{251} + \mathtt{25} \cdot y^{247} + \mathtt{F4} \cdot y^{239} + \mathtt{01} y^{223} + \mathtt{B5} \cdot y^{191} + \mathtt{8F} \cdot y^{127} + \mathtt{63}.$

Simple implementation using a 256 byte lookup table.

AES

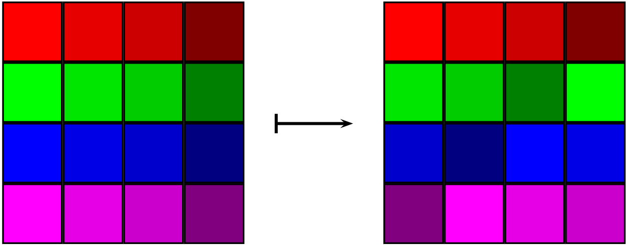
The SubBytes operation



Apply the S-box to every byte.

AES

The ShiftRows operation



The rows are shifted cyclically by zero, one, two, or three bytes.

AES

Polynomials over the field $\mathbb{F}_{2^8}$

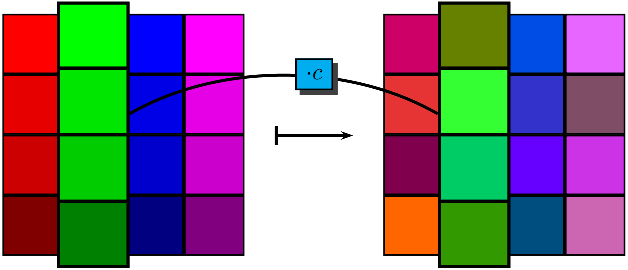
$R = \mathbb{F}_{2^8}[z]/(z^4+1) \ni a_0 + a_1 z + a_2 z^2 + a_3 z^3$,
where $a_i \in \mathbb{F}_{2^8}$.
Addition: coefficient-wise $(a+b)_i = a_i + b_i$, XOR.
Multiplication: as for polynomials modulo z^4+1. Another way to express $d = a \cdot b$ is by the following matrix equation:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Not a field: $(z+1)^4 = 0$.

AES

The MixColumns operation



Each column is considered as a polynomial and multiplied by
$c = \texttt{02} + \texttt{01}z + \texttt{01}z^2 + \texttt{03}z^3$.
Inverse: Multiply with $d = \texttt{0E} + \texttt{09}z + \texttt{0D}z^2 + \texttt{0B}z^3$.

AES

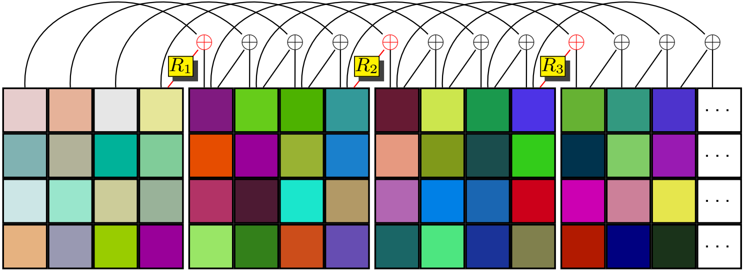
Nonlinear part of the key schedule

$$\boxed{R_i} : \begin{array}{ccc} (\mathbb{F}_{2^8})^4 & \longrightarrow & (\mathbb{F}_{2^8})^4, \\ \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} & \longmapsto & \begin{bmatrix} \boxed{S}(b) + x^{i-1} \\ \boxed{S}(c) \\ \boxed{S}(d) \\ \boxed{S}(a) \end{bmatrix} \end{array}$$

Due to the use of the S-box this map is non-linear.

AES

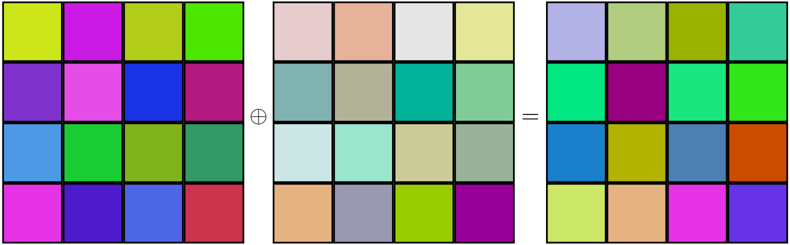
The Key Schedule



The round keys are generated from the 128 to 256 bit key.

AES

The AddRoundKey operation



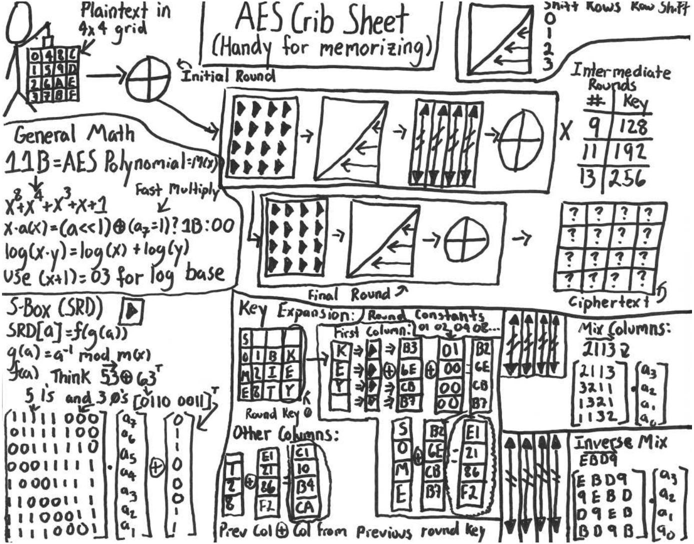
Simple XOR with the round key.

AES

- Well explained design decisions.
- Good S-box.
- Avalanche effect.
  - In one round a difference affects at least five bytes.
- Best known attack in 2015: $2^{126.1}$ steps to break AES-128 (Andrey Bogdanov, Dmitry Khovratovich & Christian Rechberger, 2012).

Practical Constructions of Block Ciphers:

AES

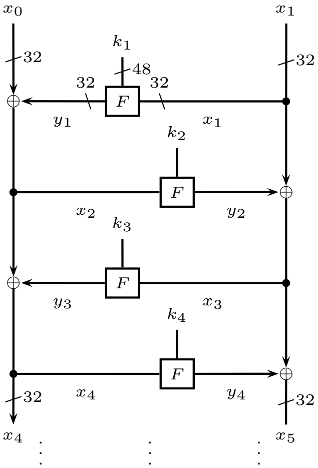


Jeff Moser (2009)

Practical Constructions of Block Ciphers:

Feistel Networks

- 1973: Basis for DES.
- Function F uses round keys, not necessarily invertible.
- Easy to invert.
- Principle reused in many variants in various ciphers.



Practical Constructions of Block Ciphers:

DES

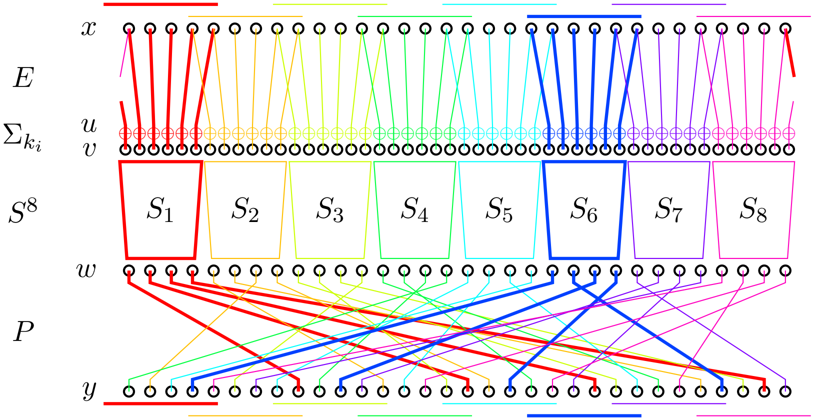


Figure: Illustration of the DES round function F_{k_i}.

Security features

- The final S-boxes have been chosen to resist differential cryptanalysis.[8]
- Avalanche effect:

  (S-4) Changing one of the six input bits of an S-box affects at least two of the four output bits.

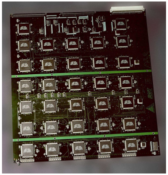
  Together with the rest of the structure that leads to a property like: Consider two 64-bit values $x^{(0)}$ and $x^{(1)}$ that differ in a single bit. Then a few rounds later all bits are affected. Namely, after eight rounds. DES uses 16 rounds.

[8] Coppersmith (1994) revealed that many years later. Actually, the original S-boxes proposed by IBM were much worse. The NSA(!) proposed the new ones and they seemingly 'knew' differential cryptanalysis.

DES broken

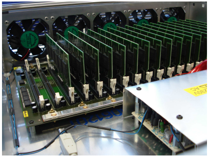
DES was designed to provide 56-bit security.

- Brute-force is practical.
  - 1998 EFF's Deep Crack (\$250 000) breaks one DES key in 56 hours.
  - 2006 Ruhr-Uni-Bochum & Uni-Kiel, COPACOBANA (\$10 000) with 120 FPGAs needs 6.4 days to break a DES key.
- Differential cryptanalysis: 2^{49} chosen plaintexts (CPA).
- Linear cryptanalysis: 2^{43} known plaintexts (KPA).



×58

EFF (2004)



Gerd Pfeiffer (2007)

Practical Constructions of Block Ciphers:

Increasing the Key Length of a Block Cipher

DES twice

- $\mathsf{Enc}_{(k_0,k_1)}(m) \leftarrow \mathsf{Enc}^{\mathsf{DES}}_{k_1}\,\mathsf{Enc}^{\mathsf{DES}}_{k_0}(m)$.
- Meet-in-the-middle: at best only 57-bit security.

DES three times, only two keys

- $\mathsf{Enc}_{(k_0,k_1)}(m) \leftarrow \mathsf{Enc}^{\mathsf{DES}}_{k_0}\,\mathsf{Dec}^{\mathsf{DES}}_{k_1}\,\mathsf{Enc}^{\mathsf{DES}}_{k_0}(m)$.
- There is an attack using 2^{56} chosen plaintexts...

3DES

- $\mathsf{Enc}^{\mathsf{3DES}}_{(k_0,k_1,k_2)}(m) \leftarrow \mathsf{Enc}^{\mathsf{DES}}_{k_2}\,\mathsf{Dec}^{\mathsf{DES}}_{k_1}\,\mathsf{Enc}^{\mathsf{DES}}_{k_0}(m)$.
- Meet-in-the-middle: at best 112-bit security.
  Not 168, but still...
- This was to be defeated by AES candidates in speed and security!

Differential cryptanalysis (Biham & Shamir, 1991)

Consider inputs x_0, x_1 with a difference Δx. Measure the amount of output y_0, y_1 with difference Δy:

$$
\begin{aligned}
&\mathsf{diff}_E(\Delta x \to \Delta y) \\
&= \mathsf{prob}\left(E(X) \oplus E(X \oplus \Delta x) = \Delta y \,\middle|\, X \xleftarrow{\text{🎲}} \{0,1\}^k \right) \\
&= \frac{1}{2^k} \# \left\{ x \in \{0,1\}^k \,\middle|\, E(x) \oplus E(x \oplus \Delta x) = \Delta y \right\} \\
&\in [0,1].
\end{aligned}
$$

If the cipher is suitably 'random' we expect this number to be small unless $\Delta x = 0$ and $\Delta y = 0$.
Any deviation should and does lead to an attack. . .

Linear cryptanalysis (Matsui 1994)

How far is the bit $\langle b \,|\, E(X)\rangle$ away from the linear function $\langle a \,|\, X\rangle$?

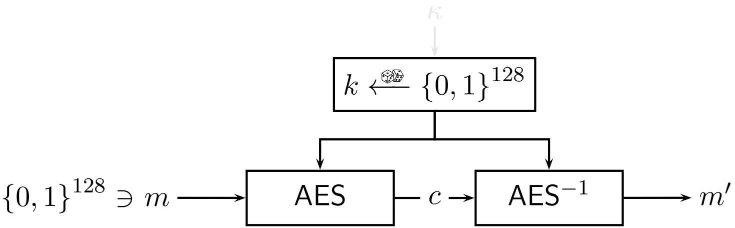
$$\begin{aligned}
&\mathsf{bias}_E(a,b) \\
&= \mathsf{prob}\Big(\langle a \,|\, X\rangle = \langle b \,|\, E(X)\rangle\Big) - \mathsf{prob}\Big(\langle a \,|\, X\rangle \neq \langle b \,|\, E(X)\rangle\Big) \\
&= 2\,\mathsf{prob}\Big(\langle a \,|\, X\rangle = \langle b \,|\, E(X)\rangle\Big) - 1 \\
&= \frac{1}{2^k} \sum_{x \in \{0,1\}^k} (-1)^{\langle a \,|\, x\rangle} (-1)^{\langle b \,|\, E(x)\rangle} \\
&\in [-1, 1].
\end{aligned}$$

If the cipher is suitably ‘random’ we expect this number to be small. Any deviation should and does lead to an attack. . .

Practical Constructions of Block Ciphers:

Summary

Advanced Encryption Standard



κ

$k \xleftarrow{\$} \{0,1\}^{128}$

$\{0,1\}^{128} \ni m$

AES

c

AES^{-1}

m'

128-bit secure. . . 126.1 remain

Practical Constructions of Block Ciphers:

Summary

We will see:

Fact

None of these block ciphers can be indistinguishable under chosen plaintext attacks.

Instead we want them to be ‘pseudorandom functions’.

Fact

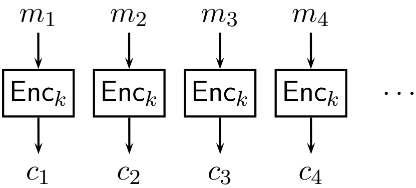
A pseudorandom function ‘is’ OW-CPA secure.

Further, these block ciphers only apply to a fixed block size. . .

Question

How can we use them for longer messages?

ECB mode

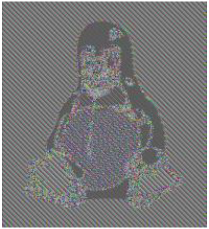


Pro:

- . . . is simple, parallelizable.
- . . . can be OW-CPA secure.

Con:

- . . . is never be indistinguishable (IND-POA secure).

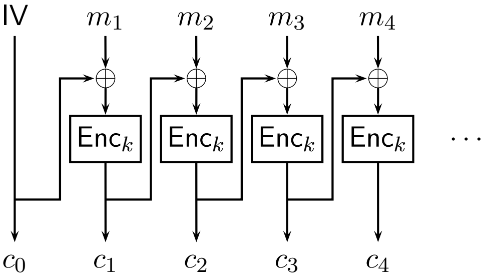


. . . (Larry Ewing, 1996)

Practical Constructions of Block Ciphers:

Modes of operation

CBC mode

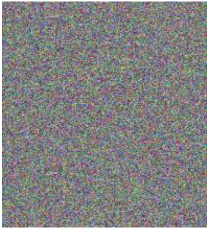


Pro:

- . . . is self synchronizing, partially parallelizable.
- . . . can be IND-CPA secure.

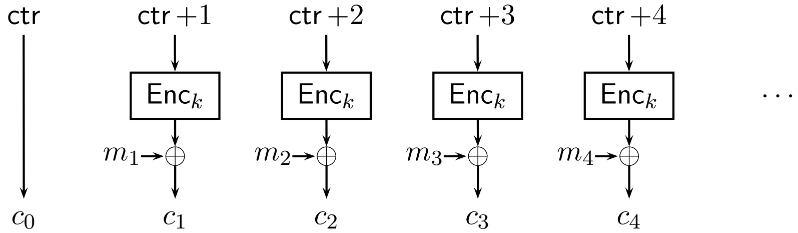
Con:

- . . . with fixed IV is not IND-CPA secure.



. . . (Larry Ewing, 1996)

CTR mode



Pro:

- . . . can be parallelized and precomputed.
- . . . can be IND-CPA secure.

Con:

- . . . is not self synchronizing.

Practical Constructions of Block Ciphers:

Modes of operation

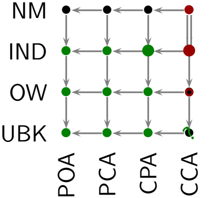
Security of these modes

Theorem

Assume that Enc.(·) *is a pseudorandom function and for all the constructions the message length is fixed. Then*

1. *ECB mode is OW-CPA secure but not IND-POA secure.*
2. *CBC- mode with a fixed initialization vector is not IND-CPA secure.*
3. *CBC mode with a random initialization vector for each message is IND-CPA secure.*
4. *CTR mode with a random initialization vector for each message is IND-CPA secure.*
5. *CTR mode with a random initialization vector for each key is IND-CPA secure.*
6. *None of these modes can be IND-CCA secure.*

We defer the detailed treatment.



See Bellare, Desai, Pointcheval & Rogaway (1998). Relations among notions of security for public-key encryption schemes.

Section 6 Overview

Symmetric-Key Encryption and Pseudorandomness, I

Practical Constructions of Block Ciphers

Symmetric-Key Encryption and Pseudorandomness, II

- Security Against Chosen-Plaintext Attacks (IND-CPA)
- Constructing CPA-secure Encryption Schemes
- Security Against Chosen-Ciphertext Attacks (IND-CCA)

MACs and Collision-Resistant Hash Functions

Symmetric-Key Encryption and Pseudorandomness, II:

Security Against Chosen-Plaintext Attacks (IND-CPA)

Indistinguishability game $G^{\textsf{IND-CPA}}$

- Prepare a key $k \leftarrow \textsf{KeyGen}(1^\kappa)$ in $\mathcal{K}$.
- Choose a hidden bit $h \xleftarrow{⚄} \{0,1\}$ uniformly random.
- Prepare an encryption oracle $\mathcal{O}_{\textsf{Enc}}$. When called with $m \in \mathcal{M}$ the oracle returns $c \leftarrow \textsf{Enc}_k(m)$.
- Prepare a one-time oracle $\mathcal{O}_{\textsf{Test}}$. When called with $m_0^*, m_1^* \in \mathcal{M}$ the oracle returns $c^* \leftarrow \textsf{Enc}_k(m_h^*)$.
- Call the attacker $\mathcal{A}$ with input 1^κ and the oracles $\mathcal{O}_{\textsf{Enc}}$ and $\mathcal{O}_{\textsf{Test}}$. Await a guess $h' \in \{0,1\}$.
- If $h = h'$ then ACCEPT else REJECT.

Definition

A symmetric-key encryption scheme Π is indistinguishable under chosen plaintext attack
iff
for each probabilistic polynomial-time attacker $\mathcal{A}$ the advantage

$$\textsf{adv}^{\textsf{IND-CPA}}(\mathcal{A}) = \left| \textsf{prob}\left(G^{\textsf{IND-CPA}}(\mathcal{A}) = \textsf{ACCEPT}\right) - \frac{1}{2} \right|$$

is negligible.

Theorem

There are symmetric-key encryption schemes that are IND-POA-secure but not IND-CPA-secure.

Theorem

A deterministic symmetric-key encryption scheme is never IND-CPA-secure.

Longer messages

Theorem (Arbitrary fixed-length length)

Given an IND-CPA-secure symmetric-key encryption scheme and fix a length μ, then the 'codebook mode' symmetric-key encryption scheme with

$$\mathsf{Enc}_k^{\mathsf{ECB}}(m_0|\ldots|m_{\mu-1}) := \mathsf{Enc}_k(m_0)|\ldots|\mathsf{Enc}_k(m_{\mu-1})$$

is also IND-CPA-secure.

Clearly, the number μ of blocks of the plaintext is clearly visible in the ciphertext. This scheme is not length-hiding.
There are schemes which are length-hiding to a certain extent.

CPA in history

WWII: Deciphering Enigma

- Placing of mines and attacks of chosen targets.
- Germar oordinates of the

WWII: Savi

- Japane ”.
- Washin nd.
- Fake m s were low.
- Japanese intercepted and reported “AF is low on water.”

⇒ Midway saved and significant losses for Japan.

Chosen plaintext attacks
are relevant!

Symmetric-Key Encryption and Pseudorandomness, II:

Security Against Chosen-Plaintext Attacks (IND-CPA)

Left-or-right game $G^{\text{LOR-CPA}}$

- Prepare a key $k \leftarrow \text{KeyGen}(1^\kappa)$ in $\mathcal{K}$.
- Choose a hidden bit $h \xleftarrow{⚄} \{0,1\}$ uniformly random.
- Prepare an oracle $\mathcal{O}_{\text{LOR}}$, called left-or-right oracle. When called with $m_0, m_1 \in \mathcal{M}$ the oracle returns $c \leftarrow \text{Enc}_k(m_h)$.
- Call the attacker $\mathcal{A}$ with input 1^κ and the oracle $\mathcal{O}_{\text{LOR}}$. Await a guess $h' \in \{0,1\}$.
- If $h = h'$ then ACCEPT else REJECT.

Definition

A symmetric-key encryption scheme Π is left-or-right-secure under chosen plaintext attack iff for each probabilistic polynomial-time attacker $\mathcal{A}$ the advantage

$$\text{adv}^{\text{LOR-CPA}}(\mathcal{A}) = \left| \text{prob}\left(G^{\text{LOR-CPA}}(\mathcal{A}) = \text{ACCEPT}\right) - \frac{1}{2} \right|$$

is negligible.

Theorem

1. *Given an IND-CPA attacker $\mathcal{A}'$ then there is an LOR-CPA attacker $\mathcal{A}$ such that*

$$\mathsf{adv}^{\mathsf{IND\text{-}CPA}}(\mathcal{A}') \leq \mathsf{adv}^{\mathsf{LOR\text{-}CPA}}(\mathcal{A}).$$

In particular: LOR-CPA secure ⇒ IND-CPA secure.

2. *Given an LOR-CPA attacker $\mathcal{A}$ that calls $\mathcal{O}_{\mathsf{LOR}}$ at most ℓ times then there is an IND-CPA attacker $\mathcal{A}'$ such that*

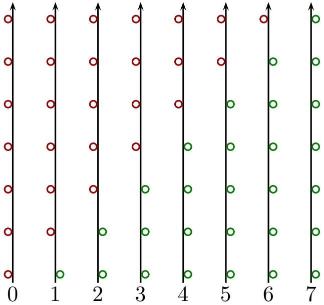
$$\mathsf{adv}^{\mathsf{LOR\text{-}CPA}}(\mathcal{A}) \leq \ell \cdot \mathsf{adv}^{\mathsf{IND\text{-}CPA}}(\mathcal{A}').$$

In particular: IND-CPA secure ⇒ LOR-CPA secure.

On the proof of (2)

Given an LOR-CPA attacker $\mathcal{A}$ construct an IND-CPA attacker $\mathcal{A}'$ as follows:

- Pick a value $t \in_{🎲} \mathbb{N}_{<\ell}$.
- Guess the hidden bit $h'' \xleftarrow{🎲} \{0, 1\}$.
- Define $\mathcal{O}_{\mathsf{LOR}}(m_0, m_1)$ as follows: On call t, return $\mathcal{O}_{\mathsf{Test}}(m_0, m_1)$, before return $\mathcal{O}_{\mathsf{Enc}}(m_{h''})$, afterwards return $\mathcal{O}_{\mathsf{Enc}}(m_{\neg h''})$.
- Call $\mathcal{A}$ and expect $h' \in \{0, 1\}$.
- Return h'.



Each column is one situation and tells which messages are encrypted by $\mathcal{O}_{\mathsf{LOR}}$ in the various calls.

A green circle to the right of the line means that $\mathcal{O}_{\mathsf{LOR}}$ uses the hidden bit h''. A red circle to the left means that it uses its complement $\neg h''$.

Pseudorandom function

- In some sense a pseudorandom function is a pseudorandom generator that outputs a function $\{0,1\}^\kappa \to \{0,1\}^\kappa$.
- The number of functions $\{\{0,1\}^\kappa \to \{0,1\}^\kappa\}$ is $2^{\ell(\kappa)}$ with $\ell(\kappa) = \kappa \cdot 2^\kappa$. This ℓ is *not* polynomial.
- A random function in $\{\{0,1\}^\kappa \to \{0,1\}^\kappa\}$ cannot be chosen or handed over at once by a polynomial time machine. But...
- We consider keyed functions

$$F\colon \begin{array}{rcl} \{0,1\}^\kappa & \longrightarrow & \{\{0,1\}^\kappa \to \{0,1\}^\kappa\}, \\ k & \longmapsto & F_k(x) \end{array}$$

with a key $k \in \{0,1\}^\kappa$.

Pseudorandom function F

Definition

A keyed function $F\colon \{0,1\}^\kappa \to \{\{0,1\}^\kappa \to \{0,1\}^\kappa\}\,,\ k \mapsto F_k(\cdot)$ is a pseudorandom function iff it is probabilistic polynomial-time computable and for each probabilistic polynomial-time distinguisher $\mathcal{D}$ the advantage

$$\mathsf{adv}_F(\mathcal{D}) = \left|\mathsf{prob}\left(\mathcal{D}(F_k(\cdot)) = 1\right) - \mathsf{prob}\left(\mathcal{D}(f(\cdot)) = 1\right)\right|$$

is negligible. Here, $k \xleftarrow{\text{⚄⚄}} \{0,1\}^\kappa$ and $f \xleftarrow{\text{⚄⚄}} \{\{0,1\}^\kappa \to \{0,1\}^\kappa\}$ are chosen uniformly at random.[9]

[9] We can choose f ad-hoc: ...

Theorem

The following are equivalent

- *One-way functions exist.*
- *Pseudorandom generators exist.*
- *Pseudorandom functions exist.*
- *Pseudorandom permutations exist.*

Pseudorandom function F, game version

$G^{\mathsf{PRF}} \colon \mathcal{D} \mapsto \{\mathsf{ACCEPT}, \mathsf{REJECT}\}$

- Pick $k \xleftarrow{🎲} \{0,1\}^{\kappa}$, $W_0 \leftarrow F_k$.
- Pick $f \xleftarrow{🎲} \{\{0,1\}^{\kappa} \to \{0,1\}^{\kappa}\}$, $W_1 \leftarrow f$.
- Choose $h^{\mathsf{PRF}} \xleftarrow{🎲} \{0,1\}$.
- Call the player $\mathcal{D}$ with input $W_{h^{\mathsf{PRF}}}$ and await its guess $h'^{,\mathsf{PRF}} \in \{0,1\}$.
- If $h^{\mathsf{PRF}} = h'^{,\mathsf{PRF}}$ then ACCEPT else REJECT.

We consider a keyed function

$$F \colon \begin{array}{ccc} \{0,1\}^{\kappa} & \longrightarrow & \{\{0,1\}^{\kappa} \to \{0,1\}^{\kappa}\}, \\ k & \longmapsto & F_k(\cdot). \end{array}$$

We compare to a random function

$$f \colon \{0,1\}^{\kappa} \longrightarrow \{0,1\}^{\kappa}.$$

A probabilistic polynomial-time attacker $\mathcal{D}$ attempts to win the game G^{PRF}. Its advantage

$$\mathsf{adv}^{\mathsf{PRF}}(\mathcal{D}) := 2\left|\mathsf{prob}\left(G^{\mathsf{PRF}}(\mathcal{D}) = \mathsf{ACCEPT}\right) - \tfrac{1}{2}\right|$$

is required to be negligible.

Encryption scheme Π_F^{try}, first try

Let $F\colon \{0,1\}^\kappa \to \{\{0,1\}^\kappa \to \{0,1\}^\kappa\}$ be a pseudorandom function.

KeyGen

Input: 1^κ.
Output: $k \in \{0,1\}^\kappa$.

- $k \xleftarrow{⚄⚄} \{0,1\}^\kappa$.

Enc

Input: k, m.
Output: c.

- $c \leftarrow F_k(m)$.

Dec

Input: k, c.
Output: m.

- $m \leftarrow F_k^{-1}(c)$.

Problems: Need permutation. And never IND-CPA secure.

Encryption scheme Π_F^{rand}, randomized

Let $F\colon \{0,1\}^{\kappa} \to \{\{0,1\}^{\kappa} \to \{0,1\}^{\kappa}\}$ be a pseudorandom function.

KeyGen

Input: 1^{κ}.
Output: $k \in \{0,1\}^{\kappa}$.

- $k \xleftarrow{\text{🎲}} \{0,1\}^{\kappa}$.

Enc

Input: k, m.
Output: c.

- Choose $r \xleftarrow{\text{🎲}} \{0,1\}^{\kappa}$.
- $c \leftarrow [r, F_k(r) \oplus m]$.

Dec

Input: k, c.
Output: m.

- $m \leftarrow F_k(c_0) \oplus c_1$.

Security

Clearly, the encryption scheme $\Pi_F^{\textsf{rand}}$ is correct and efficient.

Theorem

F pseudorandom function $\Rightarrow \Pi_F^{\textit{rand}}$ IND-CPA-secure.

Security of modes of operation

Theorem

If F is a pseudorandom function then CTR mode[10] *with F_k is IND-CPA secure.*

Exercise

Prove this.

A similar statement holds for CBC mode.

[10] with a randomly chosen initial ctr

Symmetric-Key Encryption and Pseudorandomness, II:

Security Against Chosen-Ciphertext Attacks (IND-CCA)

Indistinguishability game $G^{\text{IND-CCA}}$

- Prepare a key $k \leftarrow \text{KeyGen}(1^\kappa)$ in $\mathcal{K}$.
- Choose a hidden bit $h \xleftarrow{🎲} \{0,1\}$ uniformly random.
- Prepare an encryption oracle $\mathcal{O}_{\text{Enc}}$. When called with $m \in \mathcal{M}$ the oracle returns $c \leftarrow \text{Enc}_k(m)$.
- And prepare a decryption oracle $\mathcal{O}_{\text{Dec}}$. When called with $c \in \mathcal{C}$ the oracle returns $m \leftarrow \text{Dec}_k(c)$.
- Prepare a one-time oracle $\mathcal{O}_{\text{Test}}$. When called with $m_0^*, m_1^* \in \mathcal{M}$ the oracle returns $c^* \leftarrow \text{Enc}_k(m_h^*)$.
- Call the attacker $\mathcal{A}$ with input 1^κ and the oracles $\mathcal{O}_{\text{Enc}}$, $\mathcal{O}_{\text{Dec}}$ and $\mathcal{O}_{\text{Test}}$. Await a guess $h' \in \{0,1\}$.
- If the decryption oracle has even been called with the (first) output c^* of the test oracle as input then randomly ACCEPT or REJECT.
- If $h = h'$ then ACCEPT else REJECT.

Definition

A symmetric-key encryption scheme Π is indistinguishable under chosen ciphertext attack
iff
for each probabilistic polynomial-time attacker $\mathcal{A}$ the advantage

$$\text{adv}^{\text{IND-CCA}}(\mathcal{A}) = \left|\text{prob}\left(G^{\text{IND-CCA}}(\mathcal{A}) = \text{ACCEPT}\right) - \frac{1}{2}\right|$$

is negligible.

Symmetric-Key Encryption and Pseudorandomness, II:

Security Against Chosen-Ciphertext Attacks (IND-CCA)

Fact

Each encryption scheme seen so far is not IND-CCA secure.

Lemma

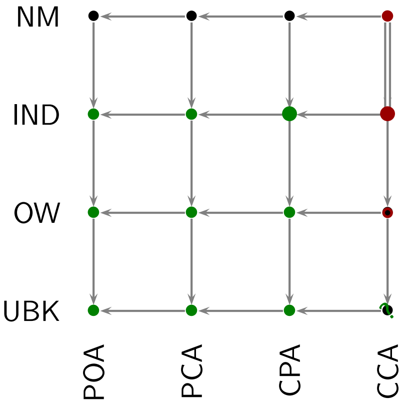
- *IND-CCA secure ⇒ IND-CPA secure.*
- *IND-CPA secure ⇒ IND-POA secure.*

Consequently, no deterministic scheme can be IND-CCA secure.

Exercise

Prove the fact for the non-deterministic schemes Π_F^{rand}.

Security landscape



Section 7 Overview

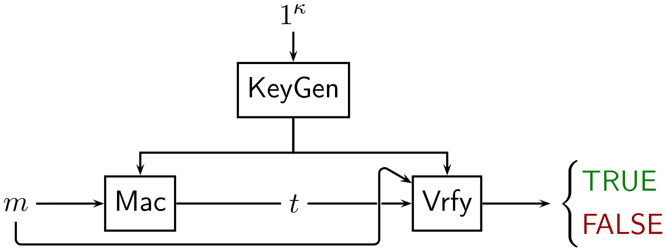
Symmetric-Key Encryption and Pseudorandomness, I

Practical Constructions of Block Ciphers

Symmetric-Key Encryption and Pseudorandomness, II

MACs and Collision-Resistant Hash Functions

- MACs — Definitions
- Constructing Secure MACs
- CBC-MAC
- *Collision-Resistant Hash Functions
- *NMAC and HMAC
- Constructing CCA-Secure Encryption Schemes
- Obtaining Privacy and Message Authentication
- AEAD, LHAE, . . .



- Correctness: $\mathsf{Vrfy}_k(m, \mathsf{Mac}_k(m)) = \mathsf{TRUE}$.
- Efficiency: probabilistic polynomial-time.
- Security: Each fast attacker has at most a small advantage in the Mac forge game G^{MAC} (see next frame).

MACs and Collision-Resistant Hash Functions:

MACs — Definitions

Mac forge game G^{MAC}

- Prepare a key $k \leftarrow \mathsf{KeyGen}(1^\kappa)$ in $\mathcal{K}$.
- Prepare a tagging oracle $\mathcal{O}_{\mathsf{MAC}}$. When called with $m \in \mathcal{M}$ the oracle returns $t \leftarrow \mathsf{Mac}_k(m)$.
- Call the attacker $\mathcal{A}$ with input 1^κ and the oracle $\mathcal{O}_{\mathsf{Mac}}$. Await a pair (m^*, t^*).
- If the tagging oracle has been called with input m^* then REJECT.
- If $\mathsf{Vrfy}_k(m^*, t^*) = \mathsf{TRUE}$ then ACCEPT else REJECT.

Definition

A (symmetric-key) message authentication scheme $\Pi = (\mathsf{KeyGen}, \mathsf{Mac}, \mathsf{Vrfy})$ is existentially unforgeable under a (adaptive) chosen-message attack (EUF-CMA secure) iff for each probabilistic polynomial-time attacker $\mathcal{A}$ the success probability

$$\mathsf{succ}^{\mathsf{MAC}}(\mathcal{A}) = \mathsf{prob}\left(G^{\mathsf{MAC}}(\mathcal{A}) = \mathsf{ACCEPT}\right)$$

is negligible.

Discussion

- Strong!
- Too much? Consider only ‘meaningful’ messages?
  No, we must have application independence.
- Replay attacks?
  The Mac does not help against these.

MACs and Collision-Resistant Hash Functions:

Constructing Secure MACs

Message Authentication Scheme $\Pi_F^{\text{mac,short}}$

Let $F\colon \{0,1\}^\kappa \to \{\{0,1\}^\kappa \to \{0,1\}^\kappa\}$ be a pseudorandom function.

KeyGen

Input: 1^κ.
Output: $k \in \{0,1\}^\kappa$.

- $k \xleftarrow{🎲} \{0,1\}^\kappa$.

Mac

Input: k, m.
Output: t.

- Return $t \leftarrow F_k(m)$.

Vrfy

Input: k, m, t.
Output: TRUE or FALSE.

- If $t = F_k(m)$ return TRUE else return FALSE.

Message Authentication Scheme $\Pi_F^{\text{mac,short}}$

Pro:

Theorem

F pseudorandom function $\Rightarrow \Pi_F^{\textit{mac,short}}$ is EUF-CMA secure.

Con:

- Works only for very short messages.

Long Message Authentication Scheme?

Options:

- ~~Use tag on XOR of message blocks.~~
  Easily broken by XORing two blocks with the same. . .
- ~~Authenticate each block separately.~~
  Easily broken by swapping two blocks. . .
- ~~Authenticate each block along with a sequence number.~~
  Easily broken by dropping final block(s). . .
- Authenticate each block along with a random id, the total length and a sequence number.
  Works!

Long Message Authentication Scheme $\Pi_F^{\text{mac,long}}$

Mac

Input: k, m.
Output: t.

- Let $\ell \leftarrow \text{length}(m)$, $d \leftarrow \left\lceil \frac{4\ell}{\kappa} \right\rceil$.
- If $\ell \geq 2^{\frac{\kappa}{4}}$ then FAIL.
- Parse $m_0 | \dots | m_{d-1} \leftarrow m | 0 \dots 0$ with $m_i \in \{0,1\}^{\frac{\kappa}{4}}$.
- Choose $r \xleftarrow{🎲} \{0,1\}^{\frac{\kappa}{4}}$.
- For $i \in \mathbb{N}_{<d}$ compute $t_i \leftarrow F_k(r|\ell|i|m_i)$ encoding $\ell, i \in \{0,1\}^{\frac{\kappa}{4}}$.
- Return $[r, t_0, \dots, t_{d-1}]$

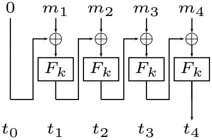
KeyGen: as before.
Vrfy: obvious.

Long Message Authentication Scheme $\Pi_F^{\text{mac,long}}$

Theorem

F pseudorandom function $\Rightarrow \Pi_F^{\textit{mac,long}}$ is EUF-CMA secure.

Fixed-length CBC-MAC $\Pi_F^{\text{cbc-mac, fixed-length}}$



Mac

Input: $k \in \{0,1\}^\kappa$, $m \in \{0,1\}^{\kappa \cdot \ell(\kappa)}$.
Output: $t \in \{0,1\}^\kappa$.

- Parse $m_0 | \ldots | m_{\ell(\kappa)-1} \leftarrow m$ with $m_i \in \{0,1\}^\kappa$.
- Let $t_0 = 0^\kappa \in \{0,1\}^\kappa$.
- For $i \in \mathbb{N}_{<d}$ compute $t_i \leftarrow F_k(t_{i-1} \oplus m_i)$.
- Return t_{d-1}.

KeyGen: as before.
Vrfy: obvious.

Fixed-length CBC-MAC $\Pi_F^{\text{cbc-mac, fixed-length}}$

Theorem

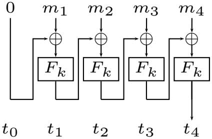
F pseudorandom function $\Rightarrow$ $\Pi_F^{\textit{cbc-mac, fixed-length}}$ is EUF-CMA secure.

- The IV t_0 is fixed. This is crucial!
- Only t_{d-1} is output. This is also crucial.
- Warning: When combining with an encryption, you must use an independent key.

Exercise

For the security of CBC-MAC and variants consider M. Bellare, J. Kilian and P. Rogaway.
The security of the cipher block chaining message authentication code. JCSS 61(3):362–399, 2000.

Variable-length CBC-MAC $\Pi_F^{\text{cbc-mac, variable-length}}$



To achieve a variable length CBC-MAC you can...

- ...use a length dependent key: $k_\ell \leftarrow F_k(\ell)$, compute the fixed-length CBC-MAC with this key.
- ...prepend the message length to the message encoded as a κ-bit string and compute the fixed-length CBC-MAC of that extended message. (Postpending is a bad idea!)
- ...derive two keys $k_1, k_2 \in \{0,1\}^\kappa$. Compute the fixed-length CBC-MAC with k_1 and return $F_{k_2}(t_{d-1})$.

Standardized variants of CBC-MAC $\Pi_F^{\text{cbc-mac}}$

- CMAC (NIST, FIPS PUB 113): XORs last (padded) block with a modified key before computing the Fixed-length CBC-MAC $\Pi_F^{\text{cbc-mac, fixed-length}}$.
- RFC 3610: specififies CCM which is AES in CTR mode plus a length-prepended CBC-MAC for messages up to $2^{64} - 1$ bytes (16 exbi bytes).[*,PDF]
- ISO/IEC 9797-1: specifies 3 paddings and 6 MAC variants.

MACs and Collision-Resistant Hash Functions:

*Collision-Resistant Hash Functions

Definition

A cryptographic(!) hash function is a collision-resistant, one-way function $h_\kappa \colon \{0,1\}^* \to \{0,1\}^{\ell(\kappa)}$. . . .

Candidates

- ~~MD5 ($\ell = 128$, collisions found)~~,
- ~~SHA-1 ($\ell = 160$, security at most 63 bits)~~,
- SHA-224 . . . SHA-512 ($\ell \in \{224, 256, 384, 512\}$),
- SHA-3 (Keccak, 1600 internal bits, $\ell \in \{224, 256, 384, 512\}$),
- BLAKE, Grøstl, JH, Skein,
- Whirlpool, RIPEMD, . . .

MACs and Collision-Resistant Hash Functions:

*NMAC and HMAC

Definition (HMAC-h)

Let h be a hash function.
We define the tag generation for HMAC-h as follows: Use the hash function on $(k \oplus \text{ipad})|m$ to otain an intermediate hash value t'. Then apply the hash function again on $(k \oplus \text{opad})|t'$ to obtain the HMAC tag t. ...

Theorem

If ... then HMAC-h is EUF-CMA secure for fixed-length messages.

Let $\Pi_E = (\mathsf{KeyGen}_E, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric-key encryption scheme and $\Pi_M = (\mathsf{KeyGen}_M, \mathsf{Mac}, \mathsf{Vrfy})$ be a message authentication code. Define EtA (Encrypt-then-Authenticate) as follows

KeyGen

Input: 1^κ.
Output: $k \in \{0,1\}^\kappa \times \{0,1\}^\kappa$.

- $k \leftarrow [\mathsf{KeyGen}_E(\kappa), \mathsf{KeyGen}_M(\kappa)]$.

Enc

Input: $[k_E, k_M]$, m.
Output: $[c, t]$.

- Compute $c \leftarrow \mathsf{Enc}_{k_E}(m)$.
- Compute $t \leftarrow \mathsf{Mac}_{k_M}(c)$.
- Return $[c, t]$.

Dec

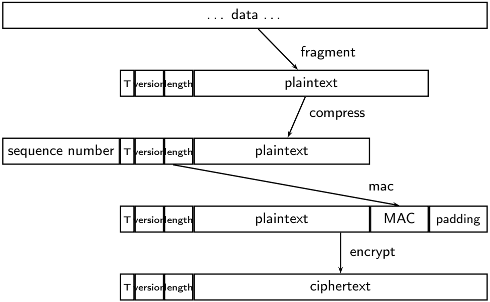
Input: $[k_E, k_M]$, $[c, t]$.
Output: m' or FAIL

- If $\mathsf{Vrfy}_{k_M}(c, t) =$ FALSE then return FAIL.
- $m' \leftarrow \mathsf{Dec}_{k_E}(c)$.
- Return m'.

MACs and Collision-Resistant Hash Functions:

Obtaining Privacy and Message Authentication

- Encrypt then Authenticate (EtA) — IPSec.
- Authenticate then Encrypt (AtE) — TLS/SSL.



- Encrypt and Authenticate (E&A) — SSH.
- ...

MACs and Collision-Resistant Hash Functions:

AEAD, LHAE, . . .

LHAE Game

Input: κ.
Output: ACCEPTor REJECT.

- $k \xleftarrow{\text{🎲}} \mathrm{AE.Keygen}(1^\kappa)$.
- $h_{\mathsf{AE}} \xleftarrow{\text{🎲}} \{0, 1\}$.
- Invoke the player with input $(\mathcal{O}_{\mathsf{enc}}, \mathcal{O}_{\mathsf{dec}})$ to obtain a bit h'_{AE}.
- If $h_{\mathsf{AE}} = h'_{\mathsf{AE}}$
  then return ACCEPT
  else return REJECT.

$$\mathsf{adv}^{\mathsf{LHAE}}(\mathcal{P}) = \left| \mathsf{prob}\left(\mathcal{P} \text{ wins the LHAE game}\right) - \frac{1}{2} \right| .$$

$\mathcal{O}_{\mathsf{enc}}$

Input: ℓ, H, m_0, m_1.
Output: c_0, c_1 or FAIL.

- $c_0 \leftarrow \mathrm{AE.Enc}(k, \ell, H, m_0)$.
- $c_1 \leftarrow \mathrm{AE.Enc}(k, \ell, H, m_1)$.
- If $c_0 =$ FAIL or $c_1 =$ FAIL then return FAIL.
- return $c_{h_{\mathsf{AE}}}$.

$\mathcal{O}_{\mathsf{dec}}$

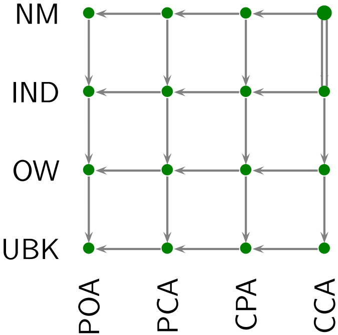
Input: H, c.
Output: m.

- If $h_{\mathsf{AE}} = 0$ then return FAIL.
- $m \leftarrow \mathrm{AE.Dec}(k, H, c)$.
- If c was created by $\mathcal{O}_{\mathsf{enc}}$ then return FAIL.
- return m.

Symmetric-Key Cryptography:

Summary

- Symmetric-key encryption, landscape.
- Practical constructions: AES, DES.
- Message authentication codes.
- IND-CCA security, authenticated encryption.



Part II

Public-Key Cryptography

Symmetric-Key Management and Public-Key Revolution

Public-Key Encryption I

Number Theory

Factoring and Computing Discrete Logarithms

Public-Key Encryption, II

*Additional Public-Key Encryption Schemes

Section 8 Overview

Symmetric-Key Management and Public-Key Revolution

- Limitations of Symmetric-Key Cryptography
- A Partial Solution — Key Distribution Centers
- Diffie-Hellman Key Exchange
- Real-or-random security
- Security and Insecurity of Diffie-Hellman Key Exchange
- The Public-Key Revolution

Public-Key Encryption I

Number Theory

Factoring and Computing Discrete Logarithms

Public-Key Encryption, II

*Additional Public-Key Encryption Schemes

Digital Signature Schemes

*Public-Key Cryptosystems in the Random Oracle Model

Problem: Key distribution

- New party joins a team: $n-1$ new keys have to be distributed. One key with each 'old' party.
- Party leaves: $n-1$ keys have to be deleted.

Partial solution: Your IT admin creates $n-1$ keys and gives one to each old party and all to the new party. But. . .

Symmetric-Key Management and Public-Key Revolution:

Limitations of Symmetric-Key Cryptography

Problem: Key storage and secrecy

- Each party must store $n-1$ secret keys.
- New party: each party must add a key to that list.
  Party leaves: each party must delete a key from the list.
- The storage must be secure!
- Some keys may be shared by many, eg. for access to a database.

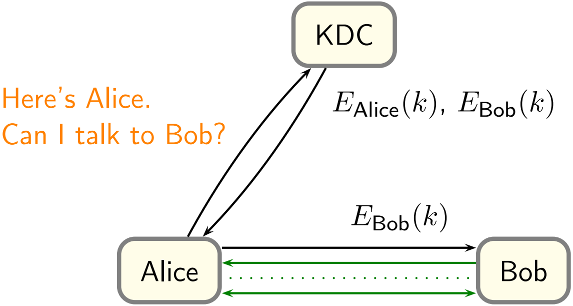
Partial solution: key distribution center. (See later.)

Problem: Open systems

- New party: possibly remote. No secret channel.

Symmetric-Key Management and Public-Key Revolution:

A Partial Solution — Key Distribution Centers



Symmetric-Key Management and Public-Key Revolution:

A Partial Solution — Key Distribution Centers

Pro

- Each party needs only a single key, namely with the KDC.
- New party:
  - Only one new key, only with KDC.
  - No other party need to act.
- Party leave: delete key at KDC.
- KDC is not locked by having to wait for Bob.

Con

- Single point of failure for safety/reliability: if KDC is offline, no connection can be started.
- Single point of failure for security: Successful attack at KDC breaks all.

A Partial Solution — Key Distribution Centers

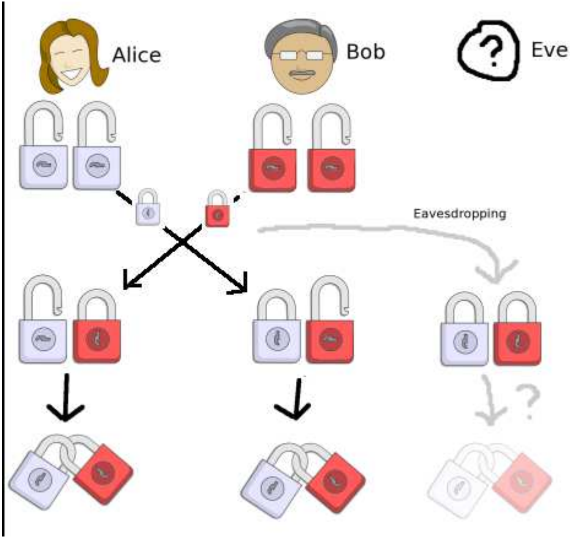
In practice:

- Needham-Schroeder protocol in the symmetric-key variant.
- Kerberos.
- Also: Needham-Schroeder protocol in the public-key variant.

Warning: IND-CPA security is not enough!

Symmetric-Key Management and Public-Key Revolution:

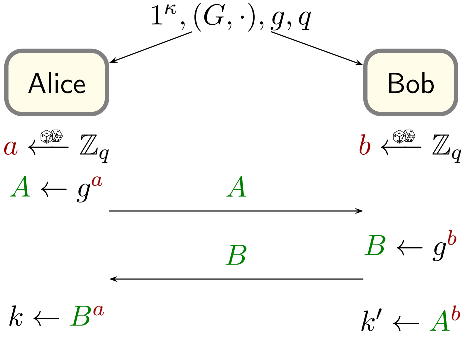
Diffie-Hellman Key Exchange



Symmetric-Key Management and Public-Key Revolution:

Diffie-Hellman Key Exchange

Based on κ fix a group $(G, \cdot)$ and an element $g \in G$ of order q.



$1^\kappa, (G, \cdot), g, q$

Alice

Bob

$a \xleftarrow{\text{🎲}} \mathbb{Z}_q$

$b \xleftarrow{\text{🎲}} \mathbb{Z}_q$

$A \leftarrow g^a$

A

$B \leftarrow g^b$

B

$k \leftarrow B^a$

$k' \leftarrow A^b$

- Correctness: $k = g^{ab} = k'$.
- Efficiency: ok, if the group operation is. Square and multiply...
- Security: ...

Security

- Necessary: the discrete logarithm problem, namely given g^x find x, is hard.
- Necessary: the Diffie-Hellman problem relative to g, namely given g^a, g^b find g^{ab}, is hard.
- Necessary: the Decisional Diffie-Hellman problem relative to g, namely given g^a, g^b, g^c decide whether $c = ab$, is hard.
- Under certain assumption. . .

Symmetric-Key Management and Public-Key Revolution:

Real-or-random security

Real-or-random game $G_{\Pi}^{\text{ROR-POA}}$

- Choose parameters $\pi \leftarrow \text{Gen}(1^\kappa)$ (mostly not randomized).
- Let Alice and Bob given the parameters π execute the key exchange protocol Π. We obtain the transcript t and the shared key k_0.
- Pick a random key $k_1 \xleftarrow{🎲} \mathcal{K}_\pi$.
- Pick a hidden bit $h \xleftarrow{🎲} \{0, 1\}$.
- Call the attacker with the parameters π, the transcript t and k_h. Await a guess $h' \in \{0, 1\}$.
- If $h' = h$ then ACCEPT else REJECT.

$$\text{adv}_{\Pi}^{\text{ROR-POA}}(\mathcal{A}) := \left|\text{prob}\left(G_{\Pi}^{\text{ROR-POA}}(\mathcal{A}) = \text{ACCEPT}\right) - \frac{1}{2}\right|$$

Definition

A key exchange Π is ROR-POA secure iff …

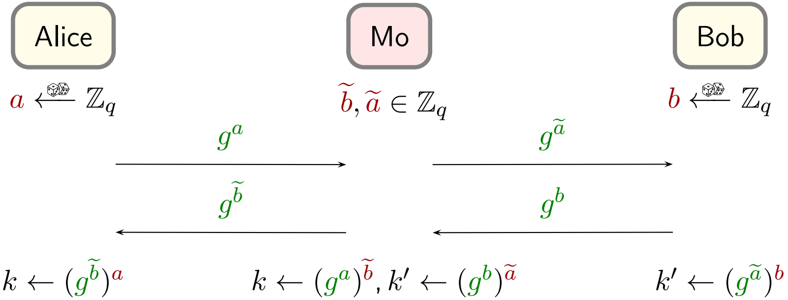
Decisional Diffie-Hellman Game

- Pick $\pi = (G, \cdot, g, q) \leftarrow \mathsf{Gen}(1^\kappa)$.
- Choose $a, b, c_1 \xleftarrow{🎲} \mathbb{Z}_q$, compute $c_0 = ab$.
- Pick a hidden bit $h \xleftarrow{🎲} \{0, 1\}$.
- Call the player with g^a, g^b, g^{c_h}. Await a guess $h' \in \{0, 1\}$.
- If $h' = h$ then ACCEPT else REJECT.

Theorem

If the Decisional Diffie-Hellman problem is hard
then the Diffie-Hellman key exchange is ROR-POA secure.

Moderator-in-the-middle



Alice

Mo

Bob

$a \xleftarrow{🎲} \mathbb{Z}_q$

$\widetilde{b}, \widetilde{a} \in \mathbb{Z}_q$

$b \xleftarrow{🎲} \mathbb{Z}_q$

g^{a}

$g^{\widetilde{a}}$

$g^{\widetilde{b}}$

g^{b}

$k \leftarrow (g^{\widetilde{b}})^{a}$

$k \leftarrow (g^{a})^{\widetilde{b}}, k' \leftarrow (g^{b})^{\widetilde{a}}$

$k' \leftarrow (g^{\widetilde{a}})^{b}$

Theorem

Basic Diffie-Hellman is never secure against an active attacker.

Symmetric-Key Management and Public-Key Revolution:

The Public-Key Revolution

CESG (1970–1974).

1970/05 ▸ Ellis (1970). The possibility of secure non-secret digital encryption.

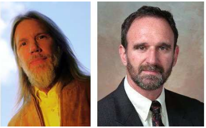
1973/11 ▸ Cocks (1973). A note on 'non-secret encryption'. [≈ RSA]

1974/01 ▸ Williamson (1974). Non-secret encryption using a finite field. [≈ DH]

Wikipe

1976/11 Diffie & Hellman (1976). New directions in cryptography.

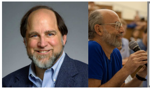
- Notion asymmetric key exchange.
- Solution: Diffie-Hellman key exchange in $\mathbb{Z}_p^\times$.
- Notion public-key encryption.
- Notion public-key signatures.



Wikipedia

1977/04 Rivest, Shamir & Adleman (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.

- Solutions for asymmetric encryption and signatures.



WikipediaWikipedia

All these systems use pairs consisting of a public and a private key.[11]

[11] Enjoy `https://www.youtube.com/watch?v=U62S8SchxX4`.

New primitives

- Public-key encryption.
- Public-key signatures.
- Interactive key exchange:

Theorem (DH, passive)

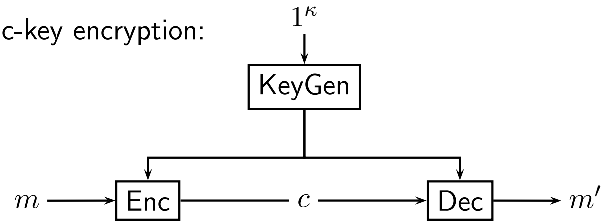
DDH hard $\Rightarrow$ Diffie-Hellman key exchange ROR-POA secure.

Theorem (DH, active)

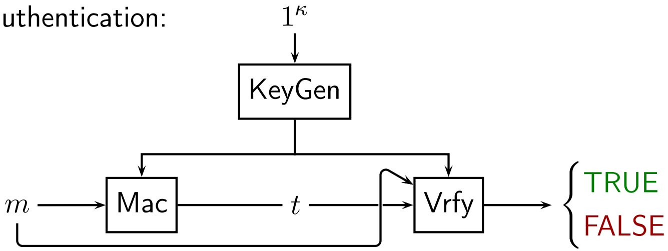
Basic Diffie-Hellman is never secure against an active attacker.

Blackbox view, symmetric

- Symmetric-key encryption:

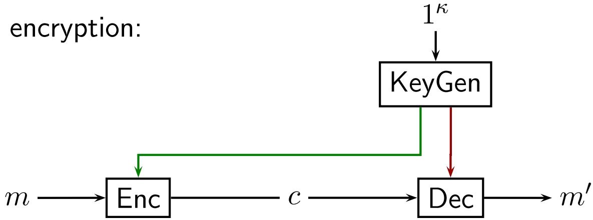


- Message authentication:

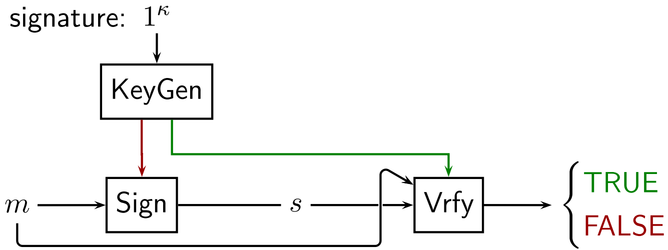


Blackbox view, asymmetric

- Public-key encryption:



- Public-key signature:



Section 9 Overview

Symmetric-Key Management and Public-Key Revolution

Public-Key Encryption I

- RSA

Number Theory

Factoring and Computing Discrete Logarithms

Public-Key Encryption, II

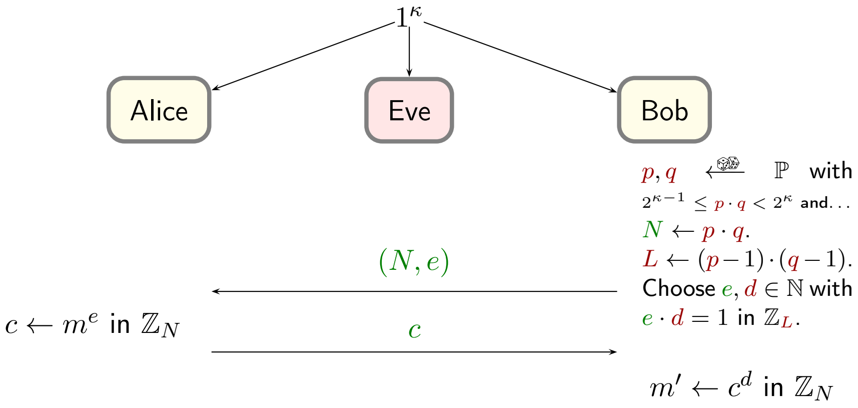
*Additional Public-Key Encryption Schemes

Digital Signature Schemes

*Public-Key Cryptosystems in the Random Oracle Model

Public-Key Encryption I:

RSA



$p, q \xleftarrow{⚄⚅} \mathbb{P}$ with $2^{\kappa-1} \leq p \cdot q < 2^\kappa$ and...
$N \leftarrow p \cdot q$.
$L \leftarrow (p-1) \cdot (q-1)$.
Choose $e, d \in \mathbb{N}$ with $e \cdot d = 1$ in $\mathbb{Z}_L$.

(N, e)

$c \leftarrow m^e$ in $\mathbb{Z}_N$

c

$m' \leftarrow c^d$ in $\mathbb{Z}_N$

Correctness: Do we always have $m' = m$?

Efficiency: Everything probabilistic polynomial-time?

Security: ??? ...

Public-Key Encryption I:

RSA

KeyGen

Input: 1^κ.

Output: A public key $(N, e) \in \mathbb{N} \times \mathbb{N}$,
a private key $(N, d) \in \mathbb{N} \times \mathbb{N}$.

- Pick $p, q \xleftarrow{⚂⚄} \mathbb{P}$ with $2^{\kappa-1} \leq p \cdot q < 2^\kappa$ and. . .
- Compute $N \leftarrow p \cdot q$.
- Compute $L \leftarrow (p - 1) \cdot (q - 1)$.
- Pick $e, d \in \mathbb{N}$ with $e \cdot d = 1$ in $\mathbb{Z}_L$.

Enc

Input: $(N, e) \in \mathbb{N} \times \mathbb{N}$,
$m \in \mathbb{Z}_N$.

Output: $c \in \mathbb{Z}_N$.

- $c \leftarrow m^e$ in $\mathbb{Z}_N$.

Dec

Input: $(N, d) \in \mathbb{N} \times \mathbb{N}$,
$c \in \mathbb{Z}_N$.

Output: $m' \in \mathbb{Z}_N$.

- $m' \leftarrow c^d$ in $\mathbb{Z}_N$.

Public-Key Encryption I:

RSA, toy example

KeyGen

Input: 1^{10}.

Output: $(N, e) = (899, 191)$, $(N, d) = (899, 431)$.

- Pick $p, q \xleftarrow{⚂⚄} \{17, 19, 23, 29, 31\}$, say $p \leftarrow 31$, $q \leftarrow 29$.
- Compute $N \leftarrow 899 = 31 \cdot 29$.
- Compute $L \leftarrow 840 = 30 \cdot 28$.
- Pick $e, d \in \mathbb{N}$ with $e \cdot d = 1$ in $\mathbb{Z}_L$. Say $e = 191$, $d = 431$.

Enc

Input: $(N, e) = (899, 191)$,
$m = 2 \in \mathbb{Z}_N$.

Output: $c \in \mathbb{Z}_N$.

- $c \leftarrow m^e = 2^{191} = 126$ in $\mathbb{Z}_{899}$.

Dec

Input: $(N, d) = (899, 431)$,
$c = 126 \in \mathbb{Z}_N$.

Output: $m' \in \mathbb{Z}_N$.

- $m' \leftarrow c^d = 126^{431} = 2$ in $\mathbb{Z}_{899}$.

Section 10 Overview

Symmetric-Key Management and Public-Key Revolution

Public-Key Encryption I

Number Theory

- Preliminaries
  - The integers $\mathbb{Z}$
  - $(G, \cdot)$ commutative group: PANIC
  - $(R, +, \cdot)$ comm. ring: PANIC+, PAN C$\cdot$, D0N^{1}T
  - Division with remainder
  - Extended Euclidean Algorithm
  - Divisibility and greatest common divisor
  - Divisibility and primes
- Modular arithmetic
  - The ring of integers modulo N
  - Inverses
  - The group $\mathbb{Z}_N^\times$ of invertible elements
  - Chinese Remainder Theorem
- Groups
  - Exponentiation
  - Exponentiation algorithm
- RSA, revisited
- Generate random primes
  - Density of primes
  - Probabilistic compositeness test
  - The Miller Rabin test
- RSA, revisited

Factoring and Computing Discrete Logarithms

Public-Key Encryption, II

*Additional Public-Key Encryption Schemes

Digital Signature Schemes

*Public-Key Cryptosystems in the Random Oracle Model

The integers $\mathbb{Z}$

- Set $\{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$, zero 0, successor $\cdot + 1$.
- Addition $a + 0 = a$, $a + (b + 1) = (a + b) + 1$, $\dots$
- Multiplication $a \cdot 0 = 0$, $a \cdot 1 = a$, $a \cdot (b + 1) = a \cdot b + a$, $\dots$

$(G, \cdot)$ commutative group: PANIC

P roperly defined: G is a set, and $\cdot\colon G \times G \to G$ is a well defined map.

A ssociative: for each $a, b, c \in G$ we have $(a \cdot b) \cdot c = a \cdot (b \cdot c)$. Computer scientist may type: $\cdot(\cdot(a, b), c) = \cdot(a, \cdot(b, c))$ considering $\cdot$ as the procedure executing the map.

N eutral element: there exists a (unique) element $1 \in G$ such that for each $a \in G$ we have $1 \cdot a = a$ and $a \cdot 1 = a$.

I nverses: for each $a \in G$ there is a (unique) $b \in G$ with $a \cdot b = 1$ and $b \cdot a = 1$.

C ommutative: for each $a, b \in G$ we have $a \cdot b = b \cdot a$.

$(G, \cdot)$ commutative group: PANIC

Examples include:

- $(\mathbb{R}, +)$, $(\mathbb{R} \setminus \{0\}, \cdot)$, $(\mathbb{Q}, +)$, $(\mathbb{Q} \setminus \{0\}, \cdot)$.
- $(\mathbb{Z}, +)$.
- $(\mathbb{Z}_N, +)$, $(\mathbb{Z}_N^\times, \cdot)$ where $N \in \mathbb{N}_{\geq 2}$.
- $(\mathbb{F}_q, +)$, $(\mathbb{F}_q^\times, \cdot)$ where q is a prime power.
- Elliptic curve groups $(E, +)$.
  - Given q an odd prime power, $a, b \in \mathbb{F}_q$ with $4a^3 + 27b^2 \neq 0$ define:
    - the set $E = \left\{[x, y] \in \mathbb{F}_q^2 \,\middle|\, y^2 = x^3 + ax + b\right\} \dot\cup \{\mathcal{O}\}$ and
    - the operation $+$ is defined such that given three distinct points P, Q, R of E on a line in $\mathbb{F}_q^2$ we have $P + Q + R = \mathcal{O}$ and $\mathcal{O}$ is the neutral element. Any line passes through $\mathcal{O}$ iff it is a vertical line.

Number Theory:

Preliminaries

$(R, +, \cdot)$ comm. ring: PANIC+, PAN C$\cdot$, D0N^{1}T

PANIC+ $(R, +)$ PANIC.

PAN C$\cdot$ $(R \setminus \{0\}, \cdot)$ PAN C.

D istributive: $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(a + b) \cdot c = a \cdot c + b \cdot c$.

0N^{1}T $0 \neq 1$.

Examples include:

- $(\mathbb{R}, +, \cdot)$, any field.
- $(\mathbb{Z}, +, \cdot)$.
- Ring $(\mathbb{Z}_N, +, \cdot)$ of integers modulo N.
- Ring $(R[x], +, \cdot)$ of univariate polynomials.

Division with remainder

Theorem

Let $a \in \mathbb{Z}$*,* $b \in \mathbb{Z}_{>0}$*. Then there exist unique integers* $q, r \in \mathbb{Z}$ *with*

- $a = q \cdot b + r$ *and*
- $0 \leq r < b$*.*

Example: $108 = 2 \cdot 42 + 24$, $0 \leq 24 < 42$.

Definition

Given $a, b \in \mathbb{Z}$, $b \neq 0$. Let $q, r \in \mathbb{Z}$ be as in the Theorem. We define

$$a \operatorname{rem} b := r.$$

Example: $108 \operatorname{rem} 42 = 24$.
Notice: $a \operatorname{rem} b \in \mathbb{Z}$.

Extended Euclidean Algorithm

Example

On input $a = 108$, $b = 42$ we fill the table

i	r_i	q_i	s_i	t_i
0	108		1	0
1	42	2	0	1
2	24	1	1	-2
3	18	1	-1	3
4	6	3	2	-5
5	0		-7	18

Definition

```
Initialize $\ell = 0$,
$r_0 = a$, $s_0 = 1$, $t_0 = 0$,
$r_1 = b$, $s_1 = 0$, $t_1 = 1$.
Until $r_{\ell+1} = 0$ repeat
    ▶ Increment $\ell$ and execute division
      with remainder $r_{\ell-1} = q_\ell r_\ell + r_{\ell+1}$.
    ▶ $s_{\ell+1} \leftarrow s_{\ell-1} - q_\ell s_\ell$.
    ▶ $t_{\ell+1} \leftarrow t_{\ell-1} - q_\ell t_\ell$.
Return $(r_\ell, s_\ell, t_\ell)$.
```

Fact

Each row has $r_i = s_i a + t_i b$.

Divisibility and greatest common divisor

- Given two numbers a, b we say that $a|b$ (a divides b) iff $\exists c$: $b = ca$.
- A number g is a greatest common divisor of two numbers a, b iff
  - it is a common divisor: $g \mid a$, $g \mid b$, and
  - any common divisor t divides it: $t \mid a \wedge t \mid b \implies t \mid g$.

Theorem

Given $a, b \in \mathbb{Z}$, $b \neq 0$. Then there exist $g, s, t \in \mathbb{Z}$ such that

$$g = sa + tb$$

and g is a greatest common divisor of a and b.

Moreover, the Extended Euclidean Algorithm outputs (g, s, t) after at most $\mathcal{O}\left(\kappa^3\right)$ bit operations[12].

[12] Actually, even within $\mathcal{O}\left(\kappa^2\right)$

Divisibility and primes

(0) A number a is invertible iff $\exists b\colon a \cdot b = 1$.

(1) A non-invertible number a is indecomposable iff in each factorization $a = b \cdot c$ (exactly) one of b, c is invertible.

(1') A non-invertible number p is prime iff $p \mid ab \;\Rightarrow\; p \mid a \vee p \mid b$.

(2+) A number a is composite iff there exists a factorization $a = b \cdot c$ with both b, c not invertible.

Lemma

- *If $c \mid ab$ and $\gcd(a, c) = 1$ then $c \mid b$.*
- *If a number p is indecomposable then p is prime.*

Theorem

If $a \mid N$ and $b \mid N$ and $\gcd(a, b) = 1$ then $ab \mid N$.

The ring of integers modulo N

For $N > 1$ we define $\mathbb{Z}_N = (\mathbb{Z}_N, +, \cdot)$ by:

- Set $\mathbb{Z}_N = \{0, 1, \ldots, N-1\} = \mathbb{Z}_{\geq 0, <N}$.
- Addition $a + b = \mathbb{Z}_N((a +_{\mathbb{Z}} b) \operatorname{rem} N)$.
- Multipliation $a \cdot b = \mathbb{Z}_N((a \cdot_{\mathbb{Z}} b) \operatorname{rem} N)$.

Definition

For $a \in \mathbb{Z}$ we define

$$a \operatorname{mod} N := \mathbb{Z}_N(a \operatorname{rem} N).$$

Notice: $a \operatorname{mod} N \in \mathbb{Z}_N$ vs. $a \operatorname{rem} N \in \mathbb{Z}$. Actually, $\operatorname{mod} N$ is a map respecting the ring structure, $\operatorname{mod} N \colon \mathbb{Z} \to \mathbb{Z}_N$.

Inverses

Theorem

Given $a, N \in \mathbb{Z}$, $N > 1$. Then

$$a \bmod N \in \mathbb{Z}_N \text{ is invertible } \iff \gcd(a, N) = 1.$$

Moreover, we can decide this and compute the inverse using the Extended Euclidean Algorithm[13]*.*

[13] Namely with input a, N, output (g, s, t) with $g = sa + tN$, $g = \gcd(a, N)$. If $g = 1$ then a is invertible with inverse s. ...

The group $\mathbb{Z}_N^\times$ of invertible elements

Definition

Define the multiplicative 'group' $\mathbb{Z}_N^\times$ of the ring $\mathbb{Z}_N$ by

- Set $\mathbb{Z}_N^\times = \{x \in \mathbb{Z}_N \,|\, x \text{ invertible}\}$.
- Operation: Multiplication $\cdot$, inherited from $\mathbb{Z}_N$.

The Euler totient function φ measures its size, $\varphi(N) := \#\mathbb{Z}_N^\times$.

Corollary

$\mathbb{Z}_N^\times = \{x \in \mathbb{Z}_N \,|\, \gcd(x, N) = 1\}$.

Fact

$\mathbb{Z}_N^\times = (\mathbb{Z}_N^\times, \cdot)$ *is a commutative group.*

Note that, given any $P, Q > 1$, $\mathbb{Z}_P \times \mathbb{Z}_Q$ is a ring.

Theorem (Chinese Remainder Theorem)

Let $N = P \cdot Q$ with $\gcd(P, Q) = 1$*. Then*

$$\begin{array}{rcl} \mathbb{Z}_N & \longrightarrow & \mathbb{Z}_P \times \mathbb{Z}_Q, \\ a \bmod N & \longmapsto & [a \bmod P, a \bmod Q] \end{array}$$

is an isomorphism respecting the ring structures.
Moreover, the inverse can be computed based on the Extended Euclidean Algorithm.[14]

[14] Namely, with input P, Q and output (g, s, t) with $g = sP + tQ$. By assumption $g = \gcd(P, Q) = 1$ and thus $1 = sP + tQ$. Noticing that $sP = 0$ in $\mathbb{Z}_P$ and $sP = 1$ in $\mathbb{Z}_Q$, we find that $(a_0, a_1) \mapsto a_0 tQ + a_1 sP$ describes the inverse map.

Number Theory:

Modular arithmetic

Example

Consider $\mathbb{Z}_{15} \cong \mathbb{Z}_3 \times \mathbb{Z}_5$:

$\mathbb{Z}_{15}$		$\mathbb{Z}_5$				
			$\mathbb{Z}_5^\times$			
		0	1	2	3	4
	0	0	6	12	3	9
$\mathbb{Z}_3$	1	10	1	7	13	4
$\mathbb{Z}_3^\times$	2	5	11	$\mathbb{Z}_{15}^\times$ 2	8	14

Algebra is respected, eg.

- invertible elements, ie. x with $\exists y\colon x \cdot y = 1$:
  $\mathbb{Z}_{15}^\times \cong \mathbb{Z}_3^\times \times \mathbb{Z}_5^\times$.
- roots of 1, ie. x with $x^2 = 1$:
  $\{1, 4, 11, 14\} \cong \{1, 2\} \times \{1, 4\}$.

Corollary

Let $N = P \cdot Q$ *with* $\gcd(P, Q) = 1$.
Then $\mathbb{Z}_N^\times \cong \mathbb{Z}_P^\times \times \mathbb{Z}_Q^\times$ *and* $\varphi(N) = \varphi(P) \cdot \varphi(Q)$.

Fact

- $\varphi(p) = p - 1$ *for* p *prime.*
- $\varphi(p \cdot q) = (p - 1) \cdot (q - 1)$ *for* p, q *distinct primes.*
- $$\varphi(N) = N \prod_{\substack{p \mid N, \\ p \text{ prime}}} \frac{p-1}{p}.$$

Note: To compute $\varphi(N)$ you need its prime divisors.

Exponentiation

Let $(G, \cdot)$ be a group, $m \in \mathbb{N}$. Then we define $g^m = 1$ iff $m = 0$ and $g^m = g^{m-1} \cdot g$ otherwise. That is,

$$g^m = \underbrace{g \cdot \ldots \cdot g}_{m \text{ times}}$$

For an additively written group $(G, +)$, we prefer to write

$$m \cdot g := \underbrace{g + \ldots + g}_{m \text{ times}}$$

Exponentiation

Theorem (Lagrange)

Consider a finite group G of size $m = \#G$ and an element $g \in G$. Then

$$g^m = 1.$$

Corollary

In the situation of the Theorem, for any $i \in \mathbb{Z}$ we have $g^i = g^{i \,\mathsf{rem}\, m}$. Consequently, we have a map

$$\mathsf{exp}_g \colon \begin{array}{rcl} \mathbb{Z}_m & \longrightarrow & G, \\ i & \longmapsto & g^i, \end{array}$$

respecting the group structure.

Exponentiation

Theorem (Euler)

Consider $N > 1$ *and an element* $g \in \mathbb{N}_{<N}$ *with* $\operatorname{gcd}(g, N) = 1$*. Then*

$$g^{\varphi(N)} = 1 \text{ in } \mathbb{Z}_N.$$

Theorem (Fermat)

Consider a prime p *and an element* $g \in \mathbb{N}$, $0 < g < p$*. Then*

$$g^{p-1} = 1 \text{ in } \mathbb{Z}_p.$$

Exponentiation algorithm

Cost of one multiplication in $\mathbb{Z}_N$ for a κ-bit integer N:

- School method: $\mathcal{O}\left(\kappa^2\right)$.
- Karatsuba: $\mathcal{O}\left(\kappa^{\log_2 3}\right) = \mathcal{O}\left(\kappa^{1.594}\right)$ [divide&conquer].
- Schönhage & Strassen (1971): $\mathcal{O}\left(\kappa \cdot \log\kappa \cdot \log\log\kappa\right)$ [FFT].
- Fürer (2007), Anindya De, Chandan Saha, Piyush Kurur and Ramprasad Saptharishi (2008): $\mathcal{O}\left(\kappa \cdot \log\kappa \cdot 2^{\log^*\kappa}\right)$.

Cost of one exponentiation in a (half) group G:

- Definition: $\#G$ multiplications in G.
- Square and multiply: $2\log_2 \#G$ multiplications in G.

Together: one exponention in $\mathbb{Z}_N$ costs at most $\mathcal{O}\left(\kappa^3\right)$.
Note: It is important that every multiplication during the exponentiation is carried out in $\mathbb{Z}_N$.

Number Theory:

RSA, revisited

With the previous we can almost completely implement RSA:

KeyGen

Input: 1^κ.
Output: $(N, e) \in \mathbb{N} \times \mathbb{N}$, $(N, d) \in \mathbb{N} \times \mathbb{N}$.

- ??? ▶ Pick $p, q \xleftarrow{\text{🎲}} \mathbb{P}$ with $2^{\kappa-1} \leq p \cdot q < 2^\kappa$ and...
- ✓$\mathcal{O}\left(\kappa^2\right)$ ▶ Compute $N \leftarrow p \cdot q$.
- ✓$\mathcal{O}\left(\kappa^2\right)$ ▶ Compute $L \leftarrow (p-1) \cdot (q-1)$.
- ✓$\mathcal{O}\left(\kappa^3\right)$ ▶ Pick $e, d \in \mathbb{N}$ with $e \cdot d = 1$ in $\mathbb{Z}_L$.

Enc

Input: $(N, e) \in \mathbb{N} \times \mathbb{N}$, $m \in \mathbb{Z}_N$.
Output: $c \in \mathbb{Z}_N$.

- ✓$\mathcal{O}\left(\kappa^3\right)$ ▶ $c \leftarrow m^e$ in $\mathbb{Z}_N$.

Dec

Input: $(N, d) \in \mathbb{N} \times \mathbb{N}$, $c \in \mathbb{Z}_N$.
Output: $m' \in \mathbb{Z}_N$.

- ✓$\mathcal{O}\left(\kappa^3\right)$ ▶ $m' \leftarrow c^d$ in $\mathbb{Z}_N$.

Number Theory:

Generate random primes

GeneratePrime

Input: 1^κ.

Output: p.

- Repeat
  - Pick a random κ-bit integer $p \xleftarrow{⚄} \mathbb{N}$.
- Until p is prime

This splits the task in two parts:

- How many iterations of the loop do we need?
- What is the cost of one prime test?

Number Theory:

Generate random primes

Density of primes

Denote by $\pi(x)$ the number of primes p with $0 < p < x$.

Theorem (Prime Number Theorem)

- *Chebyshev (1852):* $\pi(x) \sim \frac{x}{\ln x}$.
- *Schoenfeld (1976): Iff the Riemann hypothesis holds*

$$|\pi(x) - \mathrm{Li}(x)| < \frac{1}{8\pi} \ln x \textit{ for } x > 1451,$$

  where $\mathrm{Li}\, x = \int_2^x \frac{1}{\ln t}\,\mathrm{d}t \sim \frac{x}{\ln x} + \frac{x}{\ln^2 x} + 2\frac{x}{\ln^3 x}$.
- *Dusart (1998): For* $x \geq 355991$

$$\frac{x}{\ln x} + \frac{x}{\ln^2 x} + 1.8\frac{x}{\ln^3 x} < \pi(x) < \frac{x}{\ln x} + \frac{x}{\ln^2 x} + 2.51\frac{x}{\ln^3 x}.$$

Density of primes

Thus the density $\frac{\pi(x)}{x}$ of primes is roughly $\frac{1}{\ln x}$.

Corollary

The number of iterations is $\mathcal{O}(\kappa)$.

Actually, asymptotically we expect $\ln 2^{\kappa} = \kappa \ln 2$ iterations.

Probabilistic compositeness test

The bet

At an Oberwolfach meeting in the 1970s, Volker Strassen and Ernst Specker bet that a deterministic primality test will be found within ten years. The winner would be paid a ballon ride.

Probabilistic compositeness tests

- $\mathcal{O}\left(\kappa^3\right)$ ▶ Solovay & Strassen (1977).
- $\mathcal{O}\left(\kappa^3\right)$ ▶ Miller (1976), Rabin (1980).

Deterministic primality test

- $\mathcal{O}^\sim\left(\kappa^{12}\right)$ ▶ Agrawal, Kayal & Saxena (2002, +2004).
- $\mathcal{O}^\sim\left(\kappa^{6}\right)$ ▶ Many improvements. . .

AKS was too late and so Ernst Specker won the bet and the ballon ride.

Probabilistic compositeness test

- For a prime p we have $g^{p-1} = 1$ in $\mathbb{Z}_p$.
- For a composite number N the condition $g^{N-1} = 1$ in $\mathbb{Z}_N$ holds for at most $\frac{1}{4}$ of the values g.
- For primes p the polynomial equation $x^2 = 1$ has at exactly the two roots ± 1 in $\mathbb{Z}_p$.
- For a composite number the polynomial equation $x^2 = 1$ has at least four roots in $\mathbb{Z}_N$.

Conclusion

If we find a $g \in \mathbb{Z}_N$ with $g^{N-1} \neq 1$ the candidate N is not prime. If we find an element $x \in \mathbb{Z}_N$ different from ± 1 with $x^2 = 1$ the candidate N cannot be prime. And we can factor N.

The Miller Rabin test

Miller Rabin test

Input: $N \in \mathbb{N}$, $t \in \mathbb{N}$.
Output: “composite” or “maybe prime”.

- If N is even return “composite” (with factor 2).
- If N is a perfect power return “composite” (with factor).
- Write $N - 1 = 2^r u$.
- Repeat t times
  - Pick $a \xleftarrow{🎲} \mathbb{Z}_N$ and compute $[a^u, a^{2u}, a^{2^2 u}, \ldots, a^{2^r u}]$ in $\mathbb{Z}_N$.
  - If 1 is not on the list return “composite” (without factor).
  - If for some $1 \leq s \leq r$ we find $a^{2^{s-1}u} \neq \pm 1$ and $a^{2^s u} = 1$ then return “composite” (with factor).
- Return “maybe prime”.

The Miller Rabin test

Theorem

- *If p is prime then the Miller Rabin test always outputs "maybe prime".*
- *If p is composite then the Miller Rabin test outputs "composite" with probability at least $1 - 4^{-t}$.*

The Miller Rabin test needs at most $\mathcal{O}\left(t\kappa^3\right)$ bit operations to test a κ-bit number N.

Number Theory:

RSA, revisited

With the previous we can completely implement RSA:

KeyGen

Input: 1^κ.
Output: $(N, e) \in \mathbb{N} \times \mathbb{N}$, $(N, d) \in \mathbb{N} \times \mathbb{N}$.

- $\checkmark \mathcal{O}\left(\kappa^4\right)$ ▸ $\boxed{\text{Pick } p, q \xleftarrow{\text{🎲}} \mathbb{P} \text{ with } 2^{\kappa-1} \leq p \cdot q < 2^\kappa \text{ and}\ldots}$
- $\checkmark \mathcal{O}\left(\kappa^2\right)$ ▸ Compute $N \leftarrow p \cdot q$.
- $\checkmark \mathcal{O}\left(\kappa^2\right)$ ▸ Compute $L \leftarrow (p-1) \cdot (q-1)$.
- $\checkmark \mathcal{O}\left(\kappa^3\right)$ ▸ Pick $e, d \in \mathbb{N}$ with $e \cdot d = 1$ in $\mathbb{Z}_L$.

Enc

Input: $(N, e) \in \mathbb{N} \times \mathbb{N}$, $m \in \mathbb{Z}_N$.
Output: $c \in \mathbb{Z}_N$.

- $\checkmark \mathcal{O}\left(\kappa^3\right)$ ▸ $c \leftarrow m^e$ in $\mathbb{Z}_N$.

Dec

Input: $(N, d) \in \mathbb{N} \times \mathbb{N}$, $c \in \mathbb{Z}_N$.
Output: $m' \in \mathbb{Z}_N$.

- $\checkmark \mathcal{O}\left(\kappa^3\right)$ ▸ $m' \leftarrow c^d$ in $\mathbb{Z}_N$.

RSA is correct

- Recall that $N = p \cdot q$, $L = (p-1)(q-1)$.
- Encryption and decryption take place mostly in $\mathbb{Z}_N^\times$.
- Its size is $\varphi(N)$. That equals L.
- We construct e, d such that $e \cdot d = 1$ in $\mathbb{Z}_L$, ie. $d \cdot e - t \cdot L = 1$ for some $t \in \mathbb{Z}$.
- And $\mathsf{Dec}_{(N,d)}(\mathsf{Enc}_{(N,e)}(x)) = (x^e)^d = x^{ed}$ in $\mathbb{Z}_N$.
- For $x \in \mathbb{Z}_p^\times$ we know $x^{p-1} = 1$.
- Thus $x^{ed} = x^{1+tL} = x \cdot (x^{p-1})^{t(q-1)} = x$ in $\mathbb{Z}_p$.
- Also $x^{ed} = x$ is true for $x = 0 \in \mathbb{Z}_p$.
- Similarly, $x^{ed} = x$ for each $x \in \mathbb{Z}_q$.
- By the CRT then $x^{ed} = x$ in $\mathbb{Z}_N \cong \mathbb{Z}_p \times \mathbb{Z}_q$.
- Thus RSA is correct!

Theorem

RSA is correct and efficient.

Security?

- Well, if the attacker finds the primes he got it all...
- So better, that should be hard, right?
- Thus we need: Factoring is hard.

Section 11 Overview

Symmetric-Key Management and Public-Key Revolution

Public-Key Encryption I

Number Theory

Factoring and Computing Discrete Logarithms

- Factoring is hard?
- Algorithms for Factoring
  - Trial division
  - *Pollard's $p-1$ Method
  - Pollard's ϱ Method
  - Dixon's Quadratic Sieve Method
  - More and summary
- Discrete logarithm is hard?
- Algorithms for Computing Discrete Logarithms
  - Shanks' Baby-Step Giant-Step Algorithm
  - Pollard's ϱ Algorithm
  - The Pohlig & Hellman Algorithm
  - The Index Calculus Method
  - More and summary
- Recommended Key Lengths

Public-Key Encryption, II

*Additional Public-Key Encryption Schemes

Digital Signature Schemes

*Public-Key Cryptosystems in the Random Oracle Model

Factoring and Computing Discrete Logarithms:

Factoring is hard?

That is? Maybe this: Consider

~~Weak factoring experiment~~

~~Input: 1^κ.~~
~~Output: ACCEPT or REJECT.~~

- ~~Choose $\lceil \frac{\kappa}{2} \rceil$-bit integers $x_0, x_1 \xleftarrow{🎲} 2^{\lceil \frac{\kappa}{2} \rceil - 1}..2^{\lceil \frac{\kappa}{2} \rceil} - 1$.~~
- ~~Compute $N = x_0 \cdot x_1$.~~
- ~~Call the player $\mathcal{A}$ with input N and expect $x_0', x_1' \in \mathbb{N}_{>1}$.~~
- ~~If $x_0' \cdot x_1' = N$ then ACCEPT else REJECT.~~

~~and call factoring hard iff each probabilistic polynomial-time attacker $\mathcal{A}$ has at most negligible success probability.~~

Unfortunately: This is obviously wrong.
Restricting the attackers answer to $x_0', x_1' \in \mathbb{N}_{<2^{\lceil \kappa \rceil 2}}$ may be an option. But still…

Factoring and Computing Discrete Logarithms:

Factoring is hard?

~~### Full factoring experiment~~

~~Input: 1^κ.~~
~~Output: ACCEPT or REJECT.~~

- ~~Choose a κ-bit number $N \xleftarrow{\text{⚄}} \mathbb{N}$ with $2^{\kappa-1} \leq N < 2^\kappa$.~~
- ~~Call the player $\mathcal{A}$ with input N and expect its output $r, p_0, \ldots, p_{r-1}, e_0, \ldots, e_{r-1} \in \mathbb{N}$.~~
- ~~If each $\boxed{p_i \text{ is prime}}$ and $N = p_0^{e_0} \cdot p_1^{e_1} \cdot \cdots \cdot p_{r-1}^{e_{r-1}}$ then ACCEPT else REJECT.~~

~~Call factoring hard iff each probabilistic polynomial-time attacker $\mathcal{A}$ has at most negligible success.~~

Irritating point: Is this game efficient?
Yes, even deterministically with AKS.

Even more irritating: Still bad since many numbers can be factored easily:

- Primes (probability $\sim \frac{1}{\kappa \ln 2}$) or
- small multiples of primes (even more) or
- 'smooth' numbers with only very small prime divisor (few but still) or
- ...

Say GenPrimePair on 1^κ outputs a pair (p, q) of primes.

Factoring experiment relative GenPrimePair

Input: 1^κ.
Output: ACCEPT or REJECT.

- Choose primes by $(p, q) \leftarrow \text{GenPrimePair}(1^\kappa)$.
- Compute $N = p \cdot q$.
- Call the player $\mathcal{A}$ with input N and expect $p', q' \in \mathbb{N}$.
- If $p' \cdot q' = N$ then ACCEPT else REJECT.

Call factoring hard relative GenPrimePair iff each probabilistic polynomial-time attacker $\mathcal{A}$ has at most negligible success.

Trial division

- To test a κ-bit number
$$N \in \mathbb{N},$$
$2^{\kappa-1} \leq N < 2^{\kappa}$, we can try whether some number $t < N$ divides N.
- Since each divisor has a counter part $N = t \cdot t'$ and one of them is necessarily smaller than the other, we only need to consider $t \leq \sqrt{N}$.
- Each trial division takes time $\mathcal{O}\left(\kappa^2\right)$.
- Total time: $\mathcal{O}\left(2^{\frac{\kappa}{2}}\kappa^2\right) \subseteq \mathcal{O}^{\sim}\left(\sqrt{N}\right)$.

*Pollard's $p-1$ Method

Input: A number $N \in \mathbb{N}$ which is not prime and not a perfect power.
Output: A non-trivial divisor $t \in \mathbb{N}$, $t \mid N$, $1 < t < N$ or FAIL.

- Put $B \leftarrow \prod_{p \in \mathbb{P}, p < P(N)} p^{\lfloor \log_p N \rfloor}$.
- Choose $x \xleftarrow{⚄} \mathbb{Z}_N^\times$.
- $y \leftarrow x^B$ in $\mathbb{Z}_N$.
- $p \leftarrow \gcd(y-1, N)$.
- If $p \notin \{1, N\}$ then return p else return FAIL.

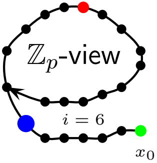
This works with a small B if for some prime divisor p of N $p-1$ is smooth, ie. $p-1$ has only small prime divisors.
If $N = p \cdot q$ for distinct primes p, q then for success we need that

- $p - 1 \mid B$ and thus $x^B = 1$ in $\mathbb{Z}_p$ but
- $q - 1 \nmid B$ and thus $x^B \neq 1$ in $\mathbb{Z}_q$ with some probability $\Omega\left(\frac{1}{\kappa}\right)$.

Conclusion: Particularly good, if $p-1$ is smooth.
Practice: Not used for crypto (but in GIMPS). But generalizes to Lenstra's elliptic curve factoring.

Pollard's (1978) ϱ Method for Factoring

- Pick numbers $x_i \xleftarrow{🎲} \mathbb{Z}_N$ until $\gcd(x_i - x_j, N)$ is non-trivial.
- Birthday paradox: about $\mathcal{O}\left(\sqrt{p}\right)$ numbers until $p \mid x_i - x_j$.
- However, we need to check all pairs which spoils all efforts.
- The ϱ: Constructing $x_{i+1} \leftarrow F(x_i)$ with some deterministic function $F \colon \mathbb{Z}_N \to \mathbb{Z}_N$, the sequence x_i must eventually 'collide' with an older x_j. In our setting that means $\gcd(x_i - x_j, N)$ is non-trivial.
  People like $F(x) = x^2 + 1$.



- But now only x_0 random: heuristic...
- Finally, Floyd's trick saves time (and memory): we only need to consider the pairs $x_{2i} = F(F(x_{2(i-1)}))$ and $x_i = F(x_{i-1})$.

Runtime: Heuristic, expected $\mathcal{O}\left(\sqrt{p}\right)$ with the smallest prime

Pollard's ϱ Method

Input: A number $N \in \mathbb{N}$.

Output: A non-trivial divisor $t \in \mathbb{N}$, $t \mid N$, $1 < t < N$
or N if it's prime or FAIL.

```
▶ If $N$ is prime return $N$.
▶ If $N$ is a perfect power return corresponding root.
▶ Pick $x_0 \xleftarrow{🎲} \mathbb{Z}_N$.
▶ $x \leftarrow x_0$, $x' \leftarrow x_0$.
▶ Repeat $\sqrt[4]{N}$ times
    ▶ $x \leftarrow F(x)$, $x' \leftarrow F(F(x'))$.
    ▶ $g \leftarrow |\gcd(x' - x, N)|$.
    ▶ If $g \notin \{1, N\}$ return $g$.
    ▶ If $g = N$ return FAIL.
▶ Return FAIL.
```

Dixon's Quadratic Sieve Method

Idea: If N is not prime, then $x^2 = 1$ has at least four solutions. Namely, by the CRT $\mathbb{Z}_N \cong \mathbb{Z}_{p_0} \times \mathbb{Z}_{p_1} \times \dots$. Then we have trivial solutions

- $(+1, +1, \dots)$, which is $+1$,
- $(-1, -1, \dots)$, which is -1,

and non-trivial solutions

- $(+1, -1, \dots)$,
- $(-1, +1, \dots)$.

Each non-trivial solution produces a proper factor of N: gcd$(x - 1, N)$.

Relaxed aim: Find $x, y \in \mathbb{Z}_N$ with $x^2 = y^2$ or $(\frac{x}{y})^2 = 1$. If this is non-trivial, ie. $x \neq \pm y$, then gcd$(x - y, N)$ is a proper factor of N.

Dixon's Quadratic Sieve Method

Observation: The elements of $\mathbb{Z}_N$ stem from elements of $\mathbb{Z}$: $\bmod N \colon \mathbb{Z} \to \mathbb{Z}_N$. And in $\mathbb{Z}$ we have unique factorization!

Let's work it out:
Relation finding: Pick some $x \in \mathbb{Z}_N$, compute $z \leftarrow x^2$ in $\mathbb{Z}_N$. Now, pull z back to $\mathbb{Z}$ and write that as a product of primes, but we only allow primes in some predetermined factor base $Q \subset \mathbb{P}$, say $Q = \{q_0, q_1, \ldots, q_{r-1}\}$. If successful, we call x good, push the factorization back to $\mathbb{Z}_N$ and obtain a relation

$$x^2 = q_0^{e_0(x)} q_1^{e_1(x)} \ldots q_{r-1}^{e_{r-1}(x)} \text{ in } \mathbb{Z}_N.$$

Well, if all exponents are even then we are done.

Dixon's Quadratic Sieve Method

Linear algebra: Given many such relations, we try to multiply some of them to yield a right hand side with only even exponents. In other words: we try to find a sum of some vectors $[e_0(x), e_1(x), \ldots, e_{r-1}(x)]^T$ that is zero modulo 2. That's a linear system with the sparse $r \times s$-matrix

$$R = \begin{bmatrix} e_0(x_0) & e_0(x_1) & e_0(x_2) & \ldots & e_0(x_{s-1}) \\ e_1(x_0) & e_1(x_1) & e_1(x_2) & \ldots & e_1(x_{s-1}) \\ \vdots & \vdots & \vdots & & \vdots \\ e_{r-1}(x_0) & e_{r-1}(x_1) & e_{r-1}(x_2) & \ldots & e_{r-1}(x_{s-1}) \end{bmatrix}$$

over the field $\mathbb{Z}_2$. If $s \gg r$ then we have a good chance that $R \cdot v = 0$ has a non-zero solution $v \in \mathbb{Z}_2^s$.

Notice: usually $s = r + 10$ is enough. So we do not need to care much about this point.

Dixon's Quadratic Sieve Method

Solving: Once v is found, we interpret $v \in \{0,1\}^s \subset \mathbb{Z}^s$ and constuct $x = \prod x_i^{v_i}$ and $e_j = \sum v_i \cdot e_j(x_i)$. Now we have a relation

$$x^2 = q_0^{e_0} q_1^{e_1} \dots q_{r-1}^{e_{r-1}} \text{ in } \mathbb{Z}_N.$$

But since v was a solution of $R \cdot v = 0$ in $\mathbb{Z}_2^s$
now all exponents e_j are even!
Thus put $y = q_0^{\frac{e_0}{2}} q_1^{\frac{e_1}{2}} \dots q_{r-1}^{\frac{e_{r-1}}{2}}$ and find

$$x^2 = y^2.$$

Heuristically, with a probability of at least $\frac{1}{2}$ we now have $x \neq \pm y$ and thus obtain a factor of N: $\gcd(x - y, N)$.

Dixon's Quadratic Sieve Method

Runtime

The two main ingredients are relation finding and linear algebra.

- Linear algebra: $\mathcal{O}\left(r^3\right)$.
- Relation finding: $(r+10) \cdot \frac{1}{\text{prob}\,(x \text{ good})} \cdot \mathcal{O}\left(r \cdot \kappa^2\right)$.

Here, x is good iff $x^2 \operatorname{rem} N$ factors over the factor base Q. Obviously, relations are easier to find if Q is larger. But then r is larger and so linear algebra is more difficult.

Balancing yields the heuristic, expected runtime

$$2^{(c+o(1))\kappa^{\frac{1}{2}}(\log_2 \kappa)^{\frac{1}{2}}}.$$

Warning: The $o\,(1)$ term hides any polynomial factor!

More and summary

As usual: the number N has κ bits and smallest prime factor p.

Algorithm	runtime
Trial division	$\mathcal{O}^{\sim}\left(2^{\frac{\kappa}{2}}\right) \subset L_{1,\frac{1}{2}}(\kappa)$
Pollard's $p-1$	$\mathcal{O}^{\sim}\left(2^{\frac{\kappa}{3}}\right)$ but...
Pollard ϱ	$\mathcal{O}^{\sim}\left(\sqrt{p}\right) \subset L_{1,\frac{1}{4}}(\kappa)$
Dixon's random squares	$L_{\frac{1}{2},\sqrt{2}}(\kappa)$
Lenstra's elliptic curve method (ECM)	$L_{\frac{1}{2},\sqrt{2}}(\log_2 p) \subset L_{\frac{1}{2},1}(\kappa)$
General number field sieve (GNFS)	$L_{\frac{1}{3},\sqrt[3]{\frac{64}{9}}}(\kappa)$
Shor's quantum factoring	$\mathrm{poly}(\kappa) = L_{0,\mathcal{O}(1)}(\kappa)$

Here, $L_{\varepsilon,c}(\kappa) = 2^{(c+o(1))\kappa^{\varepsilon} \log_2^{1-\varepsilon} \kappa}$, $\sqrt{2} = 1.41\lrcorner$, $\sqrt[3]{\frac{64}{9}} = 1.92\lrcorner$.

Factoring and Computing Discrete Logarithms:

Discrete logarithm is hard?

Say GenGroup on 1^κ outputs a triple (G, g, q) with a group G and an element $g \in G$ of order q.

Discrete logarithm experiment relative GenGroup

Input: 1^κ.
Output: ACCEPT or REJECT.

- Choose parameters $(G, g, q) \leftarrow$ GenGroup(1^κ).
- Choose $h \in \langle g \rangle = \left\{1, g, g^2, \ldots, g^{q-1}\right\}$.
- Call the player $\mathcal{A}$ with input (G, g, q), h and expect $x \in \mathbb{Z}_q$.
- If $g^x = h$ then ACCEPT else REJECT.

Call discrete logarithm hard relative GenGroup iff each probabilistic polynomial-time attacker $\mathcal{A}$ has at most negligible success.

Shanks' (1971) Baby-Step Giant-Step Algorithm

Idea: Write $x = x_1 b + x_0$ with $b = 2^{\lceil \frac{\kappa}{2} \rceil}$ and $0 \leq x_1, x_0 < b$. Solve $g^x = h$ as follows: rearrange it as

$$g^{bx_1} = h(g^{-1})^{x_0}.$$

Then construct two lists, one for each side of the equation, sort them and find the collision.

Shanks' Baby-Step Giant-Step Algorithm

Baby-Step Giant-Step

Input: G, g, q.
Output: $x \in \mathbb{Z}_q$ with $g^x = h$.

```
- $b \leftarrow \lceil \sqrt{q} \rceil$.
- For $x_1 \in \{0, \ldots, b-1\}$ add $(g^{bx_1}, x_1)$ to a list $L_0$.
- Sort the list $L_0$ wrt. the group element $g^{bx_1}$.
- For $x_0 \in \{0, \ldots, b-1\}$ do
    - Compute $s \leftarrow h(g^{-1})^{x_0}$.
    - Find $s$ is on the list $L_0$. If $s = g^{bx_1}$ then return $x_1 b + x_0$.
- Never get here.
```

Runtime: Deterministic $\mathcal{O}\left(2^{\frac{\kappa}{2}}\kappa\right)$ operations in G.

Pollard's (1978) ϱ Algorithm

Same as Pollard ϱ for factoring, only pick $a, b \xleftarrow{⚄} \mathbb{Z}_q$, compute $x_0 \leftarrow [g^a h^b, a, b]$ and proceed with

$$F\colon \begin{array}{rcl} G \times \mathbb{Z}_q \times \mathbb{Z}_q & \longrightarrow & G \times \mathbb{Z}_q \times \mathbb{Z}_q, \\ [x, a, b] & \longmapsto & \begin{cases} [g \cdot x, a+1, b], & x \in G_0, \\ [h \cdot x, a, b+1], & x \in G_1, \\ [\ \ x^2, 2a, 2b], & x \in G_2, \end{cases} \end{array}$$

for some partition $G = G_0 \mathbin{\dot\cup} G_1 \mathbin{\dot\cup} G_2$. That partition may be based on some bits of the element coding unrelated to the group structure. Given a collision, ie. $x_i = [g^a h^b, a, b]$ and $x_j = [g^{a'} h^{b'}, a', b']$ with $g^a h^b = g^{a'} h^{b'}$. Rewrite this $h^{b'-b} = g^{a-a'}$. If $b' - b$ is invertible in $\mathbb{Z}_q$ then we obtain

$$h = g^{\frac{a-a'}{b'-b}}.$$

Runtime: Heuristic, expected $\mathcal{O}^\sim\left(2^{\frac{\kappa}{2}}\right)$ operations in G.

The Pohlig & Hellman (1978) Algorithm

Idea: In case q is not prime, say $q = q' \cdot p^f$ with p prime, $f \geq 1$, we can find $x \bmod p^f$ faster: If $h = g^x$ then solve

$$h^{q'} = \left(g^{q'}\right)^x$$

determines x modulo p^f.

Notice: $\operatorname{ord}\left(g^{q'}\right) = \frac{q}{\gcd(q,q')}$.

Iterate: If $q = q'' \cdot p^e$ with $p \nmid q''$ then use the previous with $e = 1$ to find $x \bmod p$, then with $e = 2$ to find $x \bmod p^2$, then ..., until you have $x \bmod p^e$. Each of these is a discrete logarithm problem with basis $g^{\frac{q}{p}}$ whose order is p only.

CRT: Do this for all prime divisors and put the results together with the Chinese remainder theorem.

...:

The Index Calculus Method (Kraitchik 1922, Merkle 1977, Adleman 1979)

In $\mathbb{Z}_p^\times$ we can again use that this group is closely related to the ring of integers with its unique factorization.
Relation finding: Pick $x \xleftarrow{\text{⚂⚄}} \mathbb{Z}_q$ and try to write

$$g^x = q_0^{e_0(x)} \dots q_{r-1}^{e_{r-1}(x)} \text{ in } \mathbb{Z}_p^\times$$

over some fixed factor base $Q = \{q_0, \dots, q_{r-1}\}$.
Linear algebra: Solve the exponent system

$$R \cdot v = X \text{ over } \mathbb{Z}_q$$

where R's rows are $e_0(x)$, …, $e_r(x)$ and X consists of the various x to obtain the discrete logartihms of the factor base: $q_i = g^{v_i}$.

The Index Calculus Method

Solving: Pick $x \xleftarrow{🎲} \mathbb{Z}_q$ and try to write

$$hg^x = p_0^{f_0} \dots p_{r-1}^{f_{r-1}}.$$

On success obtain the wanted discrete logarithm

$$h = g^{-x+v_0 f_0 + \dots v_{r-1} f_{r-1}}.$$

Runtime: $L_{\frac{1}{2},?}(\kappa)$.
Precomputation may result in a very fast Solving!

More and summary

As usual: the number N has κ bits and smallest prime factor p.

Algorithm	runtime
Shanks' Baby-step Giant-step	$\mathcal{O}^{\sim}\left(\sqrt{q}\right) \subset L_{1,\frac{1}{2}}(\kappa)$
Pollard ϱ	$\mathcal{O}^{\sim}\left(\sqrt{q}\right) \subset L_{1,\frac{1}{2}}(\kappa)$
Pohlig & Hellman	$\mathcal{O}^{\sim}\left(\sqrt{P_{\infty}(q)}\right) \subset L_{1,\frac{1}{2}}(\kappa)$
Index calculus	$L_{\frac{1}{2},\sqrt{2}}(\kappa)$
Number field sieve (NFS)	$L_{\frac{1}{3},\sqrt[3]{\frac{64}{9}}}(\kappa)$
Joux's (2013) algorithm for very small characteristics	$L_{\frac{1}{4}+\varepsilon,c}(\kappa)$
Shor's quantum algorithm	$\mathrm{poly}(\kappa) = L_{0,\mathcal{O}(1)}(\kappa)$

Here, $L_{\varepsilon,c}(\kappa) = 2^{(c+o(1))\kappa^{\varepsilon}\log_2^{1-\varepsilon}\kappa}$, $\sqrt{2} = 1.41\lrcorner$, $\sqrt[3]{\frac{64}{9}} = 1.92\lrcorner$.

Factoring and Computing Discrete Logarithms:

Recommended Key Lengths

...from NIST (2012)

	Factoring	DL	
effective key length	RSA modulus length	order q subgroup of $\mathbb{Z}_p^\times$	Elliptic curve group order q
112	$2\,048$	p: $2\,048$, q: 224	224
128	$3\,072$	p: $3\,072$, q: 256	256
192	$7\,680$	p: $7\,680$, q: 384	384
256	$15\,360$	p: $15\,360$, q: 512	512

Compare `http://www.keylength.com/`.

Section 12 Overview

Symmetric-Key Management and Public-Key Revolution

Public-Key Encryption I

Number Theory

Factoring and Computing Discrete Logarithms

Public-Key Encryption, II

- RSA
  - *Implementation issues
  - Necessary conditions for security of RSA
  - *Attacks on misuses
- ElGamal Encryption
- Special features of RSA and ElGamal encryption
- Security for public-key encryption
- *Padded RSA
  - PKCS#1 v1.5
  - OAEP / PKCS#1 v1.20
- Public-Key Encryption — An Overview
- Definitions
- Hybrid Encryption and the KEM/DEM Paradigm
- *Trapdoor Permutations

*Additional Public-Key Encryption Schemes

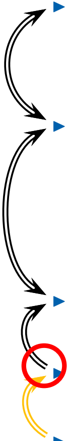
Digital Signature Schemes

*Public-Key Cryptosystems in the Random Oracle Model

*Implementation issues

- A choice has to be made for the prime generation step.
- In practice, we have to associate certain bitstrings $\{0,1\}^*$ with elements of the ring $\mathbb{Z}_N$.
- Notice that encryption is faster if e is tiny.
  - Sometimes the choice of e is restricted to a few candidates.
  - Alternatively, e may be prescribed, say $e = 2^4 + 1$, and the choice of p, q is restricted such that $\gcd(e, L) = 1$.
- And decryption is faster if d is tiny.
  - Actually, d must have $\frac{\kappa}{2}$ unpredictable bits to prevent certain attacks.
- The Chinese Remainder Theorem may be used to speed up decryption. Problems: Side channel, fault attack.
- ...

Necessary conditions for security of RSA



- Factorization is difficult: $(N, e, c) \mapsto (p, q)$.

  $\Leftarrow$ Having (p, q), we also have $L = (p-1)(q-1)$.

  $\Rightarrow$ Notice that $(x-p)(x-q) = x^2 - (N+1-L) \cdot x + N$.
  Given L we have this polynomial and thus also its roots.

- Determining L is difficult: $(N, e, c) \mapsto L$.

  $\Leftarrow$ Given L it is easy to find d.

  $\Rightarrow$ Notice that L divides $e \cdot d - 1$ in $\mathbb{Z}$.
  Given two pairs (e_0, d_0) and (e_1, d_1) we obtain L from $K := \gcd(e_0 d_0 - 1, e_1 d_1 - 1)$, since K is probably only a few bits longer than L. Given a single pair (e, d) it's complicated but possible.

- Determining d is difficult: $(N, e, c) \mapsto d = \frac{1}{e}$ in $\mathbb{Z}_L$.

  $\Leftarrow$ Given $d = \frac{1}{e}$ in $\mathbb{Z}_L$ it is easy to decrypt.

- Decryption is difficult: $(N, e, c) \mapsto m = c^d$ in $\mathbb{Z}_N$. (OW-POA.)

  $\Leftarrow$ ...

- Indistinguishability (IND-POA or better)?

*Attacks on misuses

- Encrypting short messages using tiny e.
  Insecure: m too far from uniform!
- Broadcasting using tiny e.
  Slight misuse: fixed, tiny e, same message!
- Quadratic speed up recovering small m.
  Insecure: m too far from uniform!
- Common modulus attacks.
  Misuse: N not individually chosen!
  Problems: Company knows all keys. N can be most probably be factored by any two employees. Decryption easy.

ElGamal (1985) Encryption

KeyGen

Input: 1^κ.

Output: Parameters $\pi = (G, g, q)$, a private key $a \in \mathbb{Z}_q$ and a public key $A \in G$.

- Run $(G, g, q) \leftarrow$ GenGroup(1^κ).
- Pick $a \xleftarrow{\text{⚄⚅}} \mathbb{Z}_q$, compute $A \leftarrow g^a$ in G.

Enc

Input: (π, A), $m \in G$.

Output: $c \in G \times G$.

- Pick $t \xleftarrow{\text{⚄⚅}} \mathbb{Z}_q$.
- $c \leftarrow (g^t, m \cdot A^t)$ in $G \times G$.

Dec

Input: (π, a), $c \in G \times G$.

Output: $m' \in G$.

- $m' \leftarrow c_0^{-a} \cdot c_1$ in G.

Public-Key Encryption, II:

Special features of RSA and ElGamal encryption

- RSA is deterministic.
- RSA is homomorphic:

$$\begin{aligned}
&\mathsf{Dec}_{(N,d)}\left(\mathsf{Enc}_{(N,e)}(m_0)\cdot\mathsf{Enc}_{(N,e)}(m_1)\right)\\
&=\mathsf{Dec}_{(N,d)}\left(m_0^e\cdot m_1^e\right)\\
&=\mathsf{Dec}_{(N,d)}\left((m_0\cdot m_1)^e\right) \qquad = m_0\cdot m_1.
\end{aligned}$$

- ElGamal encryption is probabilistic.
- ElGamal encryption is homomorphic:

$$\begin{aligned}
&\mathsf{Dec}_{(\pi,a)}\left(\mathsf{Enc}_{(\pi,A)}(m_0)\cdot\mathsf{Enc}_{(\pi,A)}(m_1)\right)\\
&=\mathsf{Dec}_{(\pi,a)}\left((g^{t_0},m_0\cdot A^{t_0})\cdot(g^{t_1},m_1\cdot A^{t_1})\right)\\
&=\mathsf{Dec}_{(\pi,a)}\left((g^{t_0+t_1},m_0\cdot m_1\cdot A^{t_0+t_1})\right) \qquad = m_0\cdot m_1.
\end{aligned}$$

Public-Key Encryption, II:

Security for public-key encryption

Indistinguishability game $G^{\text{IND-CPA}}$

- Pick key pair $(K, k) \leftarrow \text{KeyGen}(1^\kappa)$.
- Choose a hidden bit $h \overset{⚄}{\longleftarrow} \{0, 1\}$ uniformly random.
- Prepare an encryption oracle $\mathcal{O}_{\text{Enc}}$. When called with $m \in \mathcal{M}$ the oracle returns $c \leftarrow \text{Enc}_K(m)$.
- Prepare a one-time oracle $\mathcal{O}_{\text{Test}}$. When called with $m_0^*, m_1^* \in \mathcal{M}$ the oracle returns $c^* \leftarrow \text{Enc}_K(m_h^*)$.
- Call the attacker $\mathcal{A}$ with input 1^κ, public key K and the oracles $\mathcal{O}_{\text{Enc}}$ and $\mathcal{O}_{\text{Test}}$. Await a guess $h' \in \{0, 1\}$.
- If $h = h'$ then ACCEPT else REJECT.

Definition

A public-key encryption scheme Π is IND-CPA secure
iff
for each probabilistic polynomial-time attacker $\mathcal{A}$ the advantage

$$\text{adv}^{\text{IND-CPA}}(\mathcal{A}) =$$

$$\left| \text{prob}\left(G^{\text{IND-CPA}}(\mathcal{A}) = \text{ACCEPT} \right) - \frac{1}{2} \right|$$

is negligible.

Here, IND-POA = IND-CPA.

Is RSA IND-CPA secure?

- No, since RSA is deterministic. (Construct attacker!)

Is RSA IND-POA secure?

- No, for public-key encryption IND-POA = IND-CPA.

Is RSA OW-CPA secure?

- Yes, if(f) the RSA problem is hard, which requires essentially that RSA encryption is a one-way function.

Is ElGamal IND-CPA secure?

- Yes, if(f) DDH is hard relative to GenGroup$(\cdot)$.

Is ElGamal IND-CCA secure?

Wait, think about $G^{\text{IND-CCA}}$ first… Well, add $\mathcal{O}_{\text{Dec}}$ to $G^{\text{IND-CPA}}$.

- No, because it's homomorphic. (Construct attacker!)

Public-Key Encryption, II:

Security for public-key encryption

Indistinguishability game $G^{\text{IND-CCA}}$

- Pick key pair $(K, k) \leftarrow \text{KeyGen}(1^\kappa)$.
- Choose a hidden bit $h \overset{🎲}{\longleftarrow} \{0, 1\}$ uniformly random.
- Prepare an encryption oracle $\mathcal{O}_{\text{Enc}}$. When called with $m \in \mathcal{M}$ the oracle returns $c \leftarrow \text{Enc}_K(m)$.
- Prepare a decryption oracle $\mathcal{O}_{\text{Dec}}$. When called with $c \in \mathcal{C}$ the oracle returns $m \leftarrow \text{Dec}_k(c)$.
- Prepare a one-time oracle $\mathcal{O}_{\text{Test}}$. When called with $m_0^*, m_1^* \in \mathcal{M}$ the oracle returns $c^* \leftarrow \text{Enc}_K(m_h^*)$.
- Call the attacker $\mathcal{A}$ with input 1^κ and the oracles $\mathcal{O}_{\text{Enc}}$, $\mathcal{O}_{\text{Dec}}$ and $\mathcal{O}_{\text{Test}}$. Await a guess $h' \in \{0, 1\}$.
- If the decryption oracle has even been called with the (first) output c^* of the test oracle as input then randomly ACCEPT or REJECT.
- If $h = h'$ then ACCEPT else REJECT.

Definition

A public-key encryption scheme Π is IND-CCA secure iff for each probabilistic polynomial-time attacker $\mathcal{A}$ the advantage

$$\text{adv}^{\text{IND-CCA}}(\mathcal{A}) = \left| \text{prob}\left(G^{\text{IND-CCA}}(\mathcal{A}) = \text{ACCEPT}\right) - \frac{1}{2} \right|$$

is negligible.

IND-CCA security with short plain texts?

How to modify RSA?
Well, if the scheme prevents the attacker to use the decryption oracle on messages not produced by the encryption protocol then the attacker cannot use a modified version of the test cipher text c^*.

PKCS#1 v1.5

Idea: use some random padding before encryption.

Namely, given $2^{8(k-1)} \leq N < 2^{8k}$ preprocess $m \in \{0,1\}^{8D}$ with no zero byte as

$$\widetilde{m} = 00|02|r|00|m$$

with $r \xleftarrow{\text{⚄}} \{0,1\}^{8(k-D-3)}$.
However, this is not IND-CCA secure. Notice that the topmost bits of a valid RSA plain text $\widetilde{m}$ are known, namely 00|02. See Bleichenbacher attack, RSA hard core bit.

OAEP / PKCS#1 v1.20

Provably provides IND-CCA security provided that the RSA problem is hard.
⇒ Should be used instead of PKCS#1 v1.5.

Warning: Manger's attack on PKCS#1 v1.20

In OAEP there are two error sources but only one error message. If the implementation erroneously provides two different error messages then Manger's attack reveals the entire plaintext.

Section 13 Overview

Symmetric-Key Management and Public-Key Revolution

Public-Key Encryption I

Number Theory

Factoring and Computing Discrete Logarithms

Public-Key Encryption, II

*Additional Public-Key Encryption Schemes

- The Goldwasser-Micali Encryption Scheme
- The Rabin Encryption Scheme
- The Paillier Encryption Scheme

Digital Signature Schemes

*Public-Key Cryptosystems in the Random Oracle Model

Section 14 Overview

Symmetric-Key Management and Public-Key Revolution

Public-Key Encryption I

Number Theory

Factoring and Computing Discrete Logarithms

Public-Key Encryption, II

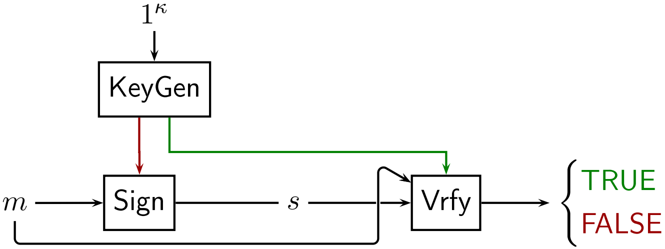
*Additional Public-Key Encryption Schemes

Digital Signature Schemes

- First schemes
  - The blackbox picture (again)
  - The “Hash-and-Sign” Paradigm
  - RSA Signature
  - ElGamal (like) Signature Scheme
- Digital Signatures — An Overview
- Definitions
- RSA Signatures
- The “Hash-and-Sign” Paradigm
- Lamport’s OneTime Signature Scheme
- *Signatures from Collision-Resistant Hashing
- ElGamal like Signatures and the DSS
- Certificates and PKIs

*Public-Key Cryptosystems in the Random Oracle Model

The blackbox picture (again)



The “Hash-and-Sign” Paradigm

For almost all schemes, the message is first hashed with a ‘cryptographically secure’ hash function $h\colon \{0,1\}^* \to D$, where $D = \{0,1\}^\kappa$, $D = \mathbb{Z}_N$ or $D = G$ as needed.

Extremes

- No hashing, ie. $h(x) = x$.
- Full-domain hash, ie. (almost) every element in D occurs as a hash (with similar probability).

RSA Signature

Signing equation: $\boxed{h(m) = s^e \text{ in } \mathbb{Z}_N}$.

KeyGen: exactly as in RSA.

Sign

Input: $(N, d) \in \mathbb{N} \times \mathbb{N}$,
$h(m) \in \mathbb{Z}_N$.

Output: $s \in \mathbb{Z}_N$.

- $s \leftarrow h(m)^d$ in $\mathbb{Z}_N$.

Verify

Input: $(N, e) \in \mathbb{N} \times \mathbb{N}$,
$m, s \in \mathbb{Z}_N$.

Output: ACCEPT or
REJECT.

- If $\boxed{h(m) = s^e \text{ in } \mathbb{Z}_N}$ then ACCEPT else REJECT.

RSA Signature

The previous scheme is known as RSA-FDH provided the used hash function h is a full domain hash function.

Theorem

If the RSA problem is hard then RSA-FDH is 'secure'.

Notice that with $h\colon \mathbb{Z}_N \to \mathbb{Z}_N$ the identity the scheme is definitely insecure.

Digital Signature Schemes:

First schemes

ElGamal (like) Signature Scheme

Signing equation: $\boxed{A^{B^*} B^c = g^{h(m)} \text{ in } G}$ with $^* \colon G \to \mathbb{Z}_q$ nice.

KeyGen: exactly as in ElGamal encryption.

Sign

Input: (π, a), $h(m) \in G$.
Output: $s \in G \times \mathbb{Z}_q$.

- Pick $b \xleftarrow{\text{🎲}} \mathbb{Z}_q^\times$.
- Compute $B \leftarrow g^b$ in G.
- Compute $c \in \mathbb{Z}_q$ such that $aB^* + bc = h(m)$.
- Return (B, c).

Verify

Input: (π, A), $m \in G$,
$s \in G \times \mathbb{Z}_q$.
Output: ACCEPT or
REJECT.

- If $\boxed{A^{B^*} B^c = g^{h(m)} \text{ in } G}$ then ACCEPT else REJECT.

ElGamal (like) Signature Scheme

Modification: Rewrite $A^{B^*} B^c = g^{h(m)}$ in G as

$$B = \left(g^{h(m)} A^{-B^*}\right)^{c^{-1}} \text{ in } G$$

and apply * on both sides. Now, we can use the signature $(B^*, c) \in \mathbb{Z}_q \times \mathbb{Z}_q$ instead of $(B, c) \in G \times \mathbb{Z}_q$.

If $G = \mathbb{Z}_p^\times$ this is much shorter in practice, compare the recommended key lengths; for example, for 128-bit security using $q \sim 2^{256}$ and $p \sim 2^{3072}$ it's only 512 bit instead of 3328 bit.

However, if G is an elliptic curve it doesn't matter.

DSA or ECDSA: is an ElGamal like signature with this modification and $G = \mathbb{Z}_p^\times$ or G an elliptic curve, respectively.

Digital Signature Schemes:

Certificates and PKIs

Certificate

A certificate is a signed electronic document with

- identification information, say a name, an email, an IP or a URL,
- one or several public keys, possibly with usage indications.

Public-Key Infrastructure (PKI)

A public-key infrastructure (PKI) consists of many certificates with the ultimate goal to grant authenticity of the final public keys.

Digital Signature Schemes:

Certificates and PKIs

. . .

Section 15 Overview

Symmetric-Key Management and Public-Key Revolution

Public-Key Encryption I

Number Theory

Factoring and Computing Discrete Logarithms

Public-Key Encryption, II

*Additional Public-Key Encryption Schemes

Digital Signature Schemes

*Public-Key Cryptosystems in the Random Oracle Model

Public-Key Cryptography:

Summary

- Diffie-Hellman and the public-key revolution.
  - Key exchange, ROR-POA, DDH, not secure against active attacker due to MitM.
- Elementary number theory.
  - Modular arithmetic. Extended Euclidean Algorithm.
- Elementary group theory.
  - Square-and-multipliy.
- RSA encryption. ElGamal encryption.
- IND-CPA, IND-CCA for public-key encryption.
- *Hybrid encryption, KEM/DEM paradigm.
- RSA signatures . . . RSA-FDH,
  ElGamal signatures . . . ECDSA.
- *EUF-CMA for public-key signatures.
- *Certificates, PKI.

Part III

Summer 2016

TAoC: The art of cryptography: secure internet & e-voting (4+2)

- Secure channels and their security.
  - IPsec, TLS, SSH, *EMV, *OTR and Open Whisper, . . .
- e-Voting, ie. remote electronic elections, anonymous channels.

SATiC: Seminar Advanced Topics in Cryptography (2)

Current research.

Master theses

Any time . . . just ask me. Some topics:
`https://cosec.bit.uni-bonn.de/students/theses/`.