Vorlesung Nr. 9

Kryptologie II - Datum: 29.10.2018

- Zufall (Teil 2)

Buch der Woche

Titel: Complexity Theory and Cryptography (2005)

Autor(en): Jörg Rothe

Verlag: Springer

Umfang: ca. 475 Seiten

Hinweise zum Inhalt:

Der erste Teil des Buches behandelt Komplexitätstheorie, der zweite Kryptographie. Beide Teile sind relativ unabhängig voneinander. Insofern ist der Buchtitel zwar richtig, doch eigentlich würde man sich einen breiteren Teil des Buches wünschen zu Komplexitätsbetrachtungen und Anwendungen innerhalb der Kryptographie. Diese kommen ebenfalls zur Sprache, nur nicht konsequent und durchgängig genug. Wer das Buch durcharbeitet, hat auf alle Fälle das notwendige Rüstzeug, um den vermissten Zusammenhängen selbst nachzugehen. Die Darstellung ist recht mathematisch und erfordert etwas Übung in Algebra und Zahlentheorie.

Kurzer Business Exkurs: Krypto Jobs...

ein paar zufällig ausgewählte Jobsuche Seiten...

Kryptographie Jobs

Sortieren nach:
Relevanz - Datum

Anstellungsart

Vollzeit	(69)
Festanstellung	(7)
Teilzeit	(6)
Befristet	(2)
Ausbildung	(1)

Ort

München	(43)
Berlin	(39)
Düsseldorf	(12)
Frankfurt am Main	(12)
Abstatt	(12)

mehr »

Unternehmen

Bosch	(36)
FERCHAU Engineering	(12)
escrypt GmbH	(11)
eSAR GmbH	(11)
exceet Secure Solutions G…	(10)

Jobs 1 - 10 von 260

Softwareentwickler (m/w) im Bereich Kryptographie und Algori...
FERCHAU Engineering GmbH ★★★☆☆ 32 Bewertungen
Bayreuth

Erarbeiten von Konzeptentwürfen im Bereich Kryptographie. Kenntnisse in den Bereichen Kryptographie, Zertifikatsmanagement und Gefahrenabwehr....

Premium-Stellenanzeige von **Ferchau** Job speichern

System-Entwickler (m/w) Kryptographie
FERCHAU Engineering GmbH ★★★☆☆ 32 Bewertungen
Friedrichshafen

Tätigkeiten in einem oder mehreren Bereichen der angewandten Kryptographie und/oder IT-Sicherheit auf verschiedenen Ebenen:....

Premium-Stellenanzeige von **Ferchau** Job speichern

Chief Technology Officer (CTO) Blockchain (m/w)
FERCHAU Engineering GmbH ★★★☆☆ 32 Bewertungen
Dresden

Kenntnisse und Interesse im Bereich Blockchaintechnologie, Smart Contract Development, Kryptographie und Visualisierung....

Premium-Stellenanzeige von **Ferchau** Job speichern

Cyber Security Tester (m/w/d)
ALPHA-ENGINEERING GmbH & Co. KG ★★★★½ 10 Bewertungen
Erlangen

Gute Kenntnisse in der Embedded Security (Tuning- / Manipulation-Protection) sowie in symmetrischer und asymmetrischer Kryptographie....

Premium-Stellenanzeige Job speichern

Programmstrategie und -koordination Digitalisierung
DLR - Deutsches Zentrum für Luft- und Raumfahrt
Köln

Cyber-Physische Systeme, Hochleistungsrechnen, High Performance Data Analytics, Maschinelles Lernen, Kryptographie, Forschungsdatenmanagement, verteilte...

vor 10 Tagen Job speichern mehr...

mehrere wissenschaftliche Mitarbeiterinnen / wissenschaftlic...
Fraunhofer-Projektgruppe Wirtschaftsinformatik des...
Augsburg

Unternehmen mit Kryptographie Jobs

CompuGroup Medical
"We are a leading eHealth company - worldwide. Our software, solutions & services support all medical and organizational activities."

Produktmanager (m/w) Software Sozialwesen

Anwendungsberater (m/w) Medizinische Software

Kaufmännischer Berater (m/w) Software IT

Jobs
Bewertungen (21)

indeed

What
Job title, keywords, or company

Cryptography

Where

Find jobs

Advanced Job Search

Tip: Enter your city or zip code in the "where" box to show results in your area.

Cryptography jobs

Sort by:
relevance - date

Salary Estimate

$90,000	(1392)
$105,000	(1172)
$115,000	(888)
$125,000	(602)
$135,000	(347)

Job Type

Full-time	(1692)
Contract	(94)
Internship	(60)
Part-time	(31)
Temporary	(19)
Commission	(6)

Location

Seattle, WA	(106)
San Francisco, CA	(94)
Annapolis Junction, MD	(91)
New York, NY	(80)
Herndon, VA	(47)

more »

Company

Amazon.com	(130)

Upload your resume - Let employers find you

Page 1 of 1,812 jobs

Don't see your role here?
Protocol Labs
Remote
Open source development, civil liberties in a digital world, network protocols, cryptography. Interested in joining Protocol Labs, but don't see the right job...
30+ days ago save job more...

Research Scientist, Cryptography
Protocol Labs
Remote
Participate in cutting-edge research in cryptography and blockchain technology. We see cryptography often acting as the last line of defense in a world that's...
30+ days ago save job more...

Applied Cryptography Engineer
Blockstack
New York, NY
Strong background in security-oriented system design, with a focus on applied cryptography. Blockstack is looking for a passionate and experienced engineer with...
25 days ago save job more...

Cybersecurity Analyst, Applied Cryptography
Visa ★★★★☆ 655 reviews
Highlands Ranch, CO
Support Data Encryption Deployments and Applied Cryptography. Support Document and evidence gathering for Applied Cryptography Audits....
25 days ago save job more...

Cryptography and Secure Computation Researcher
Galois Inc.
Portland, OR 97204 (Downtown area)
Significant fundamental or applied research focus in cryptography, and in particular, secure

Be the first to see new **Cryptography jobs**

My email:

Also get an email with jobs recommended just for me

Activate

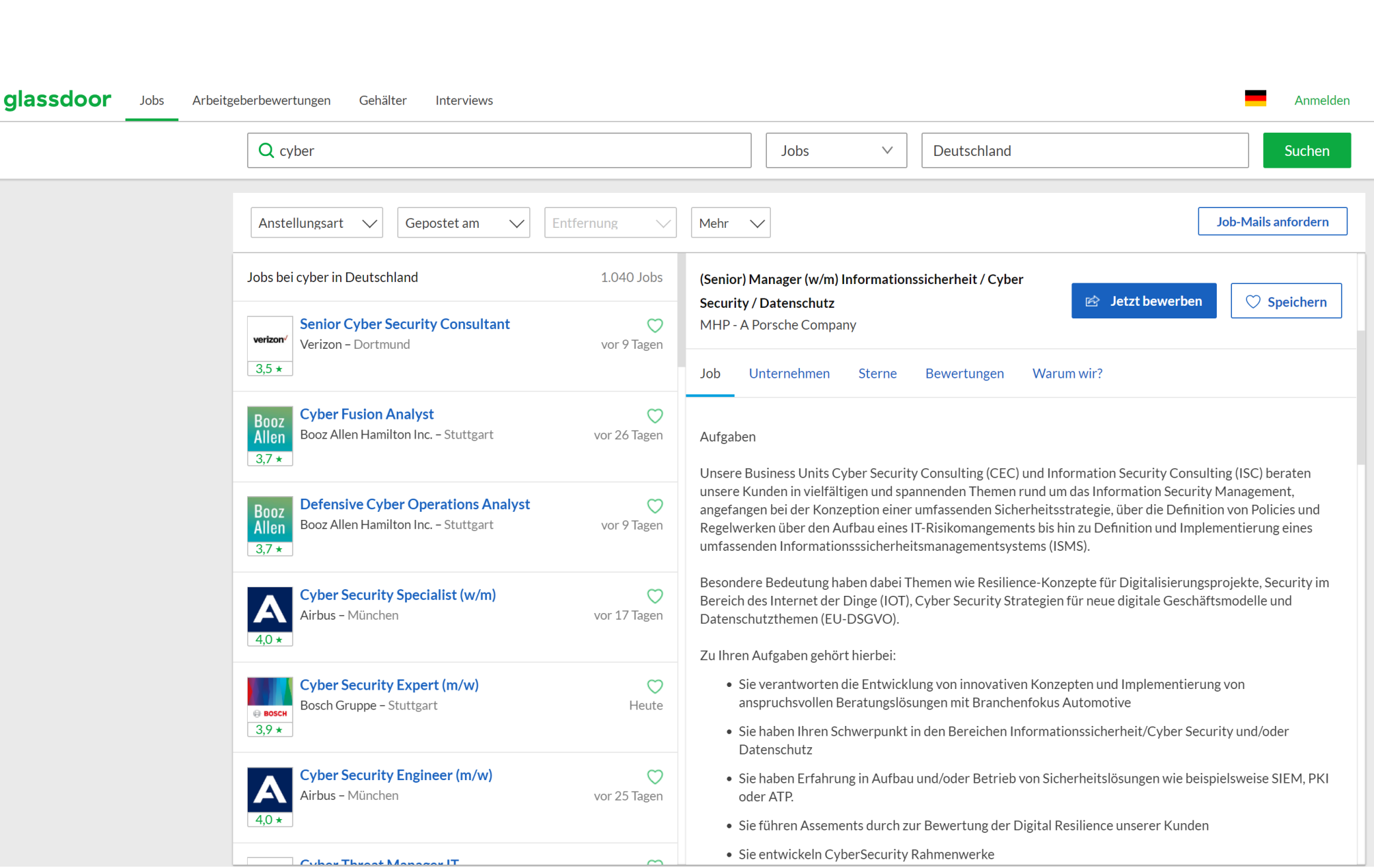
Engineer salaries in United States

$87,886 per year

Based on 20,077 salaries

Min $21,000 Max $201,000

Engineer salaries by company in United States


glassdoor
Jobs
Arbeitgeberbewertungen
Gehälter
Interviews
Anmelden
cyber
Jobs
Deutschland
Suchen
Anstellungsart
Gepostet am
Entfernung
Mehr
Job-Mails anfordern
Jobs bei cyber in Deutschland
1.040 Jobs
Senior Cyber Security Consultant
Verizon – Dortmund
3,5 ★
vor 9 Tagen
Cyber Fusion Analyst
Booz Allen Hamilton Inc. – Stuttgart
3,7 ★
vor 26 Tagen
Defensive Cyber Operations Analyst
Booz Allen Hamilton Inc. – Stuttgart
3,7 ★
vor 9 Tagen
Cyber Security Specialist (w/m)
Airbus – München
4,0 ★
vor 17 Tagen
Cyber Security Expert (m/w)
Bosch Gruppe – Stuttgart
3,9 ★
Heute
Cyber Security Engineer (m/w)
Airbus – München
4,0 ★
vor 25 Tagen
(Senior) Manager (w/m) Informationssicherheit / Cyber Security / Datenschutz
MHP - A Porsche Company
Jetzt bewerben
Speichern
Job
Unternehmen
Sterne
Bewertungen
Warum wir?
Aufgaben
Unsere Business Units Cyber Security Consulting (CEC) und Information Security Consulting (ISC) beraten unsere Kunden in vielfältigen und spannenden Themen rund um das Information Security Management, angefangen bei der Konzeption einer umfassenden Sicherheitsstrategie, über die Definition von Policies und Regelwerken über den Aufbau eines IT-Risikomangements bis hin zu Definition und Implementierung eines umfassenden Informationsssicherheitsmanagementsystems (ISMS).
Besondere Bedeutung haben dabei Themen wie Resilience-Konzepte für Digitalisierungsprojekte, Security im Bereich des Internet der Dinge (IOT), Cyber Security Strategien für neue digitale Geschäftsmodelle und Datenschutzthemen (EU-DSGVO).
Zu Ihren Aufgaben gehört hierbei:
Sie verantworten die Entwicklung von innovativen Konzepten und Implementierung von anspruchsvollen Beratungslösungen mit Branchenfokus Automotive
Sie haben Ihren Schwerpunkt in den Bereichen Informationssicherheit/Cyber Security und/oder Datenschutz
Sie haben Erfahrung in Aufbau und/oder Betrieb von Sicherheitslösungen wie beispielsweise SIEM, PKI oder ATP.
Sie führen Assements durch zur Bewertung der Digital Resilience unserer Kunden
Sie entwickeln CyberSecurity Rahmenwerke

Cryptography

Stadt, Bundesland, Land ...

Jobs finden

Geben Sie einen Standort ein: Frankfurt am Main, HE ?

Datum ▾ Suchen in ▾ Firma ▾ Quelle ▾ Umkreis ▾

52 Jobs - Seite 1 von 4 (0.04 Sekunden)

ⓘ Wussten Sie schon, dass neuvoo Ihnen die **vielfältigste** Jobauswahl in Deutschland bietet?

Cloud Solutions Security Engineer (m/f)

EPLAN Software & Service GmbH & Co. KG | Monheim am Rhein, Nordrhein-Westfalen - v...

And security protocols as well as **cryptography** Experienced with Docker or Microsoft Azure Very good language skills in English and optimally German or Polish (depends on location) Eplan Software...

IT Project Manager (m/f) **Cryptography** – Chief Security Office

Deutsche Bank | Eschborn, Hessen - vor 2 Tg.

Deutsche Bank db.com careers Analysis Discover global opportunities for responsible minds at Deutsche Bank Sometimes you have to look closely to bring synergy to our business areas...

Cryptography Systems Expert (m/f) – Chief Security … | Eschborn nah...

Senior Engineer (m/f) - Chief Security Office (CSO) / … | Eschborn, Hes...

Cryptography Expert (m/f) CISO Security Architecture… | Eschborn, Hes...

Assistent (m/w) Project Management Office - Informat… | Eschborn, Hes...

Cryptography Engineer - PKI & HSM

Excelerate. | Germany - vor 16 Std.

Job Ref. JT1050. Salary. 750 Per Day. Benefits. N A. Start Date. ASAP. Overview. **Cryptography** Engineer... Responsibilities for the **Cryptography** Engineer. PKI & HSM will include (but not limited to). Provide 3...

Blum Blum Shub (BBS)

Einer der bekanntesten für kryptographische Zwecke geeigneten Pseudozufallsgeneratoren ist der ***Blum Blum Shub Generator***, benannt nach seinen drei Erfindern. Der Algorithmus funktioniert folgendermaßen:

1. Wähle zwei große Primzahlen p und q mit $p \equiv q \equiv 3 \bmod 4$; setze $n = pq$
2. Wähle zufälliges $s \in \mathbb{N}$ mit $1 < s < n$, so dass s und n teilerfremd sind
3. Erzeuge eine Bitfolge $b_1, b_2, \ldots$ wie folgt:

   $a_0 = s^2 \bmod n$

   $a_{i+1} = a_i^2 \bmod n$

   $b_i = a_i \bmod 2$

   Wir quadrieren also einfach nur fortgesetzt modulo n und entnehmen in jedem Schritt jeweils das Least significant bit.

Der BBS Generator wurde ausführlich analysiert und seine Sicherheit hat offenkundig Bezüge zum Faktorisierungsproblem. Es lässt sich zeigen, dass das "Knacken" von BBS mindestens so schwierig ist wie das Berechnen quadratischer Reste, wobei letzteres ein in der Kryptographie anerkannt schwieriges Problem darstellt. Weitere Details lassen wir an dieser Stelle aus, da wir zuvor die erforderliche Zahlentheorie bereitstellen müssten.

Übungsaufgaben (1)

Aufgabe 9.1

Suchen Sie sich (mindestens) eine Linux Live Distribution aus, die vom USB Stick gestartet wird und die für besonders schützenswerte Anwendungsszenarien erstellt wurde (beispielsweise zum Schutz der Anonymität).

Für mögliche Kandidaten (andere Distros sind ebenso erlaubt) siehe beispielsweise

https://www.techradar.com/news/best-linux-distro-privacy-security

https://fossbytes.com/secure-linux-distros-privacy-anonymity/

Betrachten Sie den für die Erzeugung von Pseudozufallszahlen und kryptographischen Schlüsseln verwendeten Entropy Pool des Systems zum Startzeitpunkt, unmittelbar nach dem Bootvorgang. Vergleichen Sie dessen Inhalt, nachdem Sie die Live-Distribution wiederholt starten und/oder den USB-Stick auf einem anderen Host-Computer starten. Besteht eine Gefahr, dass der Initialzustand jeweils ähnlich oder gar gleich ist und folglich identische Pseudozufallsstrings erzeugt werden könnten?

Übungsaufgaben (2)

Aufgabe 9.2

Für symmetrische Verschlüsselung gibt es AES als Standard, für asymmetrische Verschlüsselung gibt es beispielsweise RSA, für Hashfunktionen SHA-2. Finden Sie heraus, ob es in analoger Weise auch Standards (im Sinne allgemein anerkannter Industrie-Standards und/oder behördlicher Standards) gibt für die Erzeugung von

(a) echten Zufallszahlen und von

(b) Pseudozufallszahlen.

Übungsaufgaben (3)

Aufgabe 9.3

Search source code repositories (e.g. github etc.) on the internet and try to find at least one software application that uses a “wrong” (pseudo)random number generator, i.e. something like Python’s random function for the generation of cryptographic keys or nonces.

Describe your search parameters (how did you find this code example, e.g. Google search strings) and explain your findings. Where is the problem, what is the problem, what are the consequences?

Abgabe der Übungsaufgaben

Deadline für die Abgabe der Übungsaufgaben 9.1 bis 9.3 ist Sonntag, der 11. November 2018 bis 23:59 Uhr.

Randomness in Python and in Programming

/dev/random versus /dev/urandom

In Linux, if we need cryptographic pseudorandom numbers, we can get them either from a file named /dev/urandom or from /dev/random

Both get their input from the same CSPRNG.

Nevertheless, there are some differences in behaviour between /dev/urandom and /dev/random:

- /dev/random tries to estimate how much entropy is left in the entropy pool. If it doesn't seem to be enough, then /dev/random stops providing any output.
- /dev/urandom will not make this check and will always continue to output pseudorandom bits.

On some systems there is a third alternative, /dev/arandom This only blocks if during an initialization phase (namely, the boot process) there is still not enough entropy available. Later, /dev/arandom never blocks and behaves like /dev/urandom.

Advantages and Disadvantages of /dev/random and /dev/urandom

In Internet blogs, many people argue about using /dev/random or /dev/urandom.

Many experts say, “just use /dev/urandom, don’t worry and forget about /dev/random”.

Indeed, both get the same input. Moreover, after proper initialization, in most cases it is safe to rely on /dev/random, even if a regular reseed is not possible because the entropy pool doesn’t get enough fresh input.

If /dev/random blocks, this can result in a denial of service condition.

Some applications, e.g. if we wipe a hard disk and write random bits on it, this can take a very long time due to an “entropy bottleneck” and not due to slow write operations.

If a “new” system still couldn’t generate enough entropy to seed the entropy pool, it could indeed result in problems when using /dev/urandom.

For long term cryptographic keys (i.e. not session keys but personal encryption keys), use either /dev/random or use /dev/urandom on a properly seeded system.

getrandom() in Linux

For details about getrandom() see Linux man pages and the Linux Programmer's Manual.

Excerpt from Linux documentation:

> *"The kernel random-number generator relies on entropy gathered from device drivers and other sources of environmental noise to seed a cryptographically secure pseudorandom number generator (CSPRNG). It is designed for security, rather than speed.*
>
> *The following interfaces provide access to output from the kernel CSPRNG: The /dev/urandom and /dev/random devices, both described in random(4). …."*

Remark: for security reasons, sometimes we install applications "changerooted" (see also chroot in Linux) to prevent them from access to sensitive file paths and data. In that case, it won't be possible to access the required file paths for /dev/urandom and /dev/random. However, getrandom() doesn't require the use of path names and/or file descriptors. Therefore, getrandom() works in changerooted situations as well.

Question: could an attacker use this nonavailability of the "real" /dev/random and /dev/urandom to feed false "random" data into such an environment?

os.urandom in Python

If we do not want to rely on Python's pseudorandom number generation, we can simply ask the operating system for help. Python's os module contains the function ***os.urandom()*** for this purpose. This works for both, Windows and Linux-type operating systems.

os.urandom(k) returns a string with k bytes of random bits, suitable for cryptographic applications.

One disadvantage of os.urandom() is that the quality of our return values depends on what the underlying operating system will provide.

In case of Windows, it uses CryptGenRandom() (details not published by Microsoft)

In case of Linux, it mostly uses the getrandom() syscall in so-called blocking mode, which means it blocks until the urandom entropy pool is initialized and contains 128 bits of entropy provided by the kernel.

Remark: Python changed the implementation details in version 3.5, again in 3.5.2, again in 3.6, and probably so in the future again. It is important to always do some research and check all the details of your current environment! Portability and upward/downward compatibility are also an issue. For that reason, it might be better to stay within Python without operating system specific functions.

os.urandom Example

Let’s look at an example of how to use os.random().

Example:

```
>>> import os
>>> os.urandom(8)
b'\x95\\\xc8\x1b}L\x8d\x8d'
>>> os.urandom(20)
b'~,\xe4\x873$\xb1\xec\x05\x88&\x14_\xcexU\x83\x07T\xeb’
```

Remark: as we can see, the output looks a little bit weird. This has nothing to do with os.urandom. We can get 256 different output bytes, and only some of them represent ASCII-printable characters. Thus, Python shows printable characters where possible (e.g. ‘T’) and uses hexadecimal (e.g. ‘\xe4’) interpretation for the remaining bytes. The b means “bytes” and the content is included in `''`

os.urandom Example

For better clarity, let's see more examples:

Examples:

```
>>> os.urandom(1)
b'\xcc'
>>> os.urandom(1)
b'N'
>>> os.urandom(1)
b'u'
>>> os.urandom(1)
b'\xe6'
>>> os.urandom(1)
b'8'
>>>
```

random Module in Python

Python provides a module named random. This contains several statistically different pseudorandom number generators for various (also non-uniform) distributions and applications. It is supposed to be used for modelling and simulation. Most of the random module's functions use Python's basic function ***random()***.

random() has the following properties:

- it generates 53-bit precision floating-point numbers
- $0 \leq \text{random}(x) < 1$
- the sequence of numbers to be generated has a period of $2^{19937-1}$
- it is implemented in C
- it uses an algorithm called Mersenne Twister
- its output is deterministic and should not be used for cryptographic applications

Many software developers don't read the documentation and use random() to generate cryptographic keys. If, for example, you google source code of security-relevant software components, you'll find more than one example…

If random() isn't secure, what else can we use?

secrets Module in Python (1)

If we need cryptographically strong pseudorandom numbers, Python's ***secrets*** module is much better than the random module described before.

Example: choose an arbitrary element from a given sequence:

```
>>> import secrets
>>> secrets.choice('0,1')
'0'
>>> secrets.choice('0,1')
','
```

As we can see, the comma was part of the sequence here. If we only want to choose between 0 and 1 we need to use

```
>>> secrets.choice('01')
‘1'
>>>
```

secrets Module in Python (2)

Example: choose an arbitrary integer i with $0 \leq i < n$ (in our case, $n = 2^{128}$):

```
>>> secrets.randbelow(2**128)

103431691270056106221275800012839070343

>>>
```

Example: choose an arbitrary integer i with n random bits (in our case, n = 2048):

```
>>> secrets.randbits(2048)

178232034987585852531063444399114198492771523223271870418330756585529599265467
360687705957134233675885602401020898493250239693568273618540570464202993899490
763757859945365530905251935579527543433259787621616425688490383672064787043729
638979859269325204110482749676408127085159753679008726416746784944987469263529
762476193861406416720816798590454934837275064308711369847642269751379743329062
052106642269752516139757447338680998753309913947731332931734375332548044274919
465991465257294255827740452603599210141406209976912891032184722758569536636188
524121947894962179829702107568681184732566953218956207405677522676992 72

>>>
```

Example: generate n arbitrary bytes, where each byte is converted into two hexadecimal digits (in our example, n = 3):

```
>>> secrets.token_hex(3)

'd8735c'
```

secrets Module in Python (3)

In web application development, we often have to include random strings in URLs to avoid token guessing attacks. Python's secrets module already contains a function for that: **secrets.token_urlsafe**

Example: generate a random string of length n, containing only URL-safe text characters in Base64 encoding:

```
>>> secrets.token_urlsafe(4)
>>> 'yxaGqw'
```

Remark: note that with secrets.token_urlsafe(n) we don't necessarily get n characters in return. Since Base64 converts three octets into four characters, on average we get 1.3 characters per byte. More general, secrets.token_urlsafe(n) returns approximately 1.3 × n characters in total.

We get back to Base64 when studying block ciphers in more detail...

Index	Char	Index	Char	Index	Char	Index	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

source: Wikipedia

More PRNGs you shouldn’t use

Python’s random function isn’t the only one that should be avoided in cryptographic applications. Other languages have similar candidates. They are all non-cryptographic. Nevertheless, you will find them in more than one key generation scenario...

- libc: rand and drand48
- PHP: rand and mt_rand
- Ruby: Random
- Python: random