Blockchiffren Teil 2

Krypto II

VL 12

12.11.2018

Buch der Woche

Titel: Cryptography for Dummies

Autor(en): Chey Cobb

Verlag: Wiley (2004)

Umfang: ca. 300 Seiten

Hinweise zum Inhalt:

Die Serie “for Dummies” umfasst so gut wie alle Bereiche der IT. Je nach Autor ist die Darstellung des jeweiligen Themas sehr trivial oder sehr verständlich oder beides zugleich. Im vorliegenden Fall ist ein guter Kompromiss gelungen. Unseren Ansprüchen an einen Hochschulkurs werden die Inhalte zwar nicht gerecht. Der Vorteil des Buches liegt jedoch darin, auch Themen aufzugreifen, die wir aus Zeitgründen nicht besprochen haben. Insofern liefert es eine nette Ergänzung als “Coffee table book” zum Schmökern am Feierabend und um sich Anregungen für tiefer gehende Betrachtungen der dargestellten Themen zu holen. Insgesamt durchaus zu empfehlen, aber nur zur Vorbereitung und/oder Ergänzung anspruchsvollerer Fachliteratur.

National Initiative for Cybersecurity Education (NICE) <csrc.nist@service.govdelivery.com>

New Data Show Demand for Cybersecurity Professionals Accelerating

Having trouble viewing this email? View it as a Web page.

NICE
NATIONAL INITIATIVE FOR CYBERSECURITY EDUCATION

New Data Show Demand for Cybersecurity Professionals Accelerating

Employer demand for cybersecurity professionals across the United States continues to accelerate, according to new data published this week on CyberSeek™.

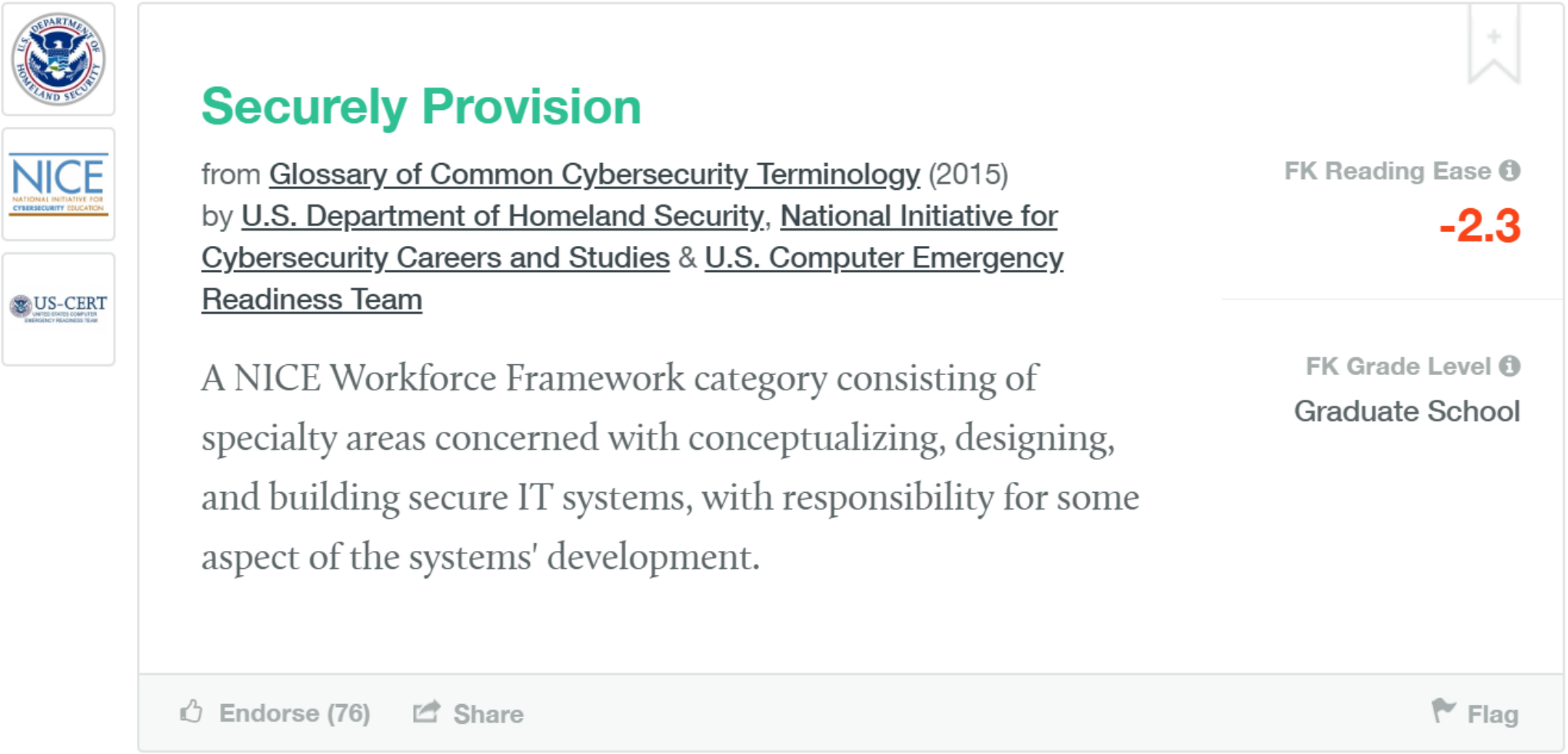
U.S. employers in the private and public sectors posted an estimated 313,735 job openings for cybersecurity workers between September 2017 and August 2018. That's in addition to the 715,000-plus cybersecurity workers currently employed around the country.

CyberSeek is aligned with the National Institute for Standards and Technology (NIST's) NICE Cybersecurity Workforce Framework, which categorizes and describes cybersecurity work and helps CyberSeek provide clear data on which job roles are most in demand. The latest CyberSeek update reveals that positions in Operate and Maintain (207,190 openings), Securely Provision (186,864), Protect and Defend (129,716), and Analyze (124,389) are the most sought after by employers.

Read the full press release on www.nist.gov.

Expore CyberSeek at www.CyberSeek.org.

Mit Abstand größter Bedarf innerhalb der Cyber Security Stellenangebote:



Securely Provision

from Glossary of Common Cybersecurity Terminology (2015)
by U.S. Department of Homeland Security, National Initiative for Cybersecurity Careers and Studies & U.S. Computer Emergency Readiness Team

A NICE Workforce Framework category consisting of specialty areas concerned with conceptualizing, designing, and building secure IT systems, with responsibility for some aspect of the systems' development.

FK Reading Ease
-2.3

FK Grade Level
Graduate School

Endorse (76) Share Flag

Kurzer Exkurs vorab: Aktueller Bericht vom 5.11.2018 zu Schwachstellen selbstverschlüsselnder Disks

U

US-CERT <US-CERT@ncas.us-cert.gov>

Self-Encrypting Solid-State Drive Vulnerabilities

Homeland Security

US-CERT United States Computer Emergency Readiness Team

National Cyber Awareness System:

Self-Encrypting Solid-State Drive Vulnerabilities

11/06/2018 07:17 PM EST

Original release date: November 06, 2018

NCCIC is aware of reports of vulnerabilities in the hardware encryption of certain self-encrypting solid-state drives. An attacker could exploit these vulnerabilities to obtain access to sensitive information.

NCCIC encourages users and administrators to review Microsoft's Security Advisory ADV180028 and Samsung's Customer Notice regarding Samsung SSDs for more information and refer to vendors for appropriate patches and recommendations, when available.

Consumer Notice regarding Samsung SSDs

In light of recent reporting for potential breach of self-encrypting SSDs in the case of expert's physical possession and specific technical settings, Samsung provides the following options of added protection for our valued consumers:

For non-portable SSDs:

We recommend installing encryption software (freeware available online) that is compatible with your system.

For portable SSDs:

We recommend updating the firmware on your device. Firmware patch can be updated through Portable SSD Activation Software. For T5 and T3 products, you must first reinstall Portable SSD Activation Software (Version 1.6.2), provided on the Samsung SSD Customer Support page (URL below), before updating the firmware. Please visit the following website for Samsung SSD Customer Support page:
http://www.samsung.com/semiconductor/minisite/ssd/download/tools/

For updating the firmware on T1 products, please contact the nearest Samsung Service Center.
Please visit the following website for contact information of Samsung Service Centers around the globe:
https://www.samsung.com/semiconductor/minisite/ssd/support/cs/

ADV180028 | Guidance for configuring BitLocker to enforce software encryption

Security Advisory

Published: 11/06/2018

Microsoft is aware of reports of vulnerabilities in the hardware encryption of certain self-encrypting drives (SEDs). Customers concerned about this issue should consider using the software only encryption provided by BitLocker Drive Encryption™. On Windows computers with self-encrypting drives, BitLocker Drive Encryption™ manages encryption and will use hardware encryption by default. Administrators who want to force software encryption on computers with self-encrypting drives can accomplish this by deploying a Group Policy to override the default behavior. Windows will consult Group Policy to enforce software encryption only at the time of enabling BitLocker.

To check the type of drive encryption being used (hardware or software):

1. Run 'manage-bde.exe -status' from elevated command prompt.
2. If none of the drives listed report "Hardware Encryption" for the **Encryption Method** field, then this device is using software encryption and is not affected by vulnerabilities associated with self-encrypting drive encryption.

For drives that are encrypted using a vulnerable form of hardware encryption, you can mitigate the vulnerability by switching to software encryption using Bitlocker with a Group Policy.

Note: After a drive has been encrypted using hardware encryption, switching to software encryption on that drive will require that the drive be unencrypted first and then re-encrypted using software encryption. If you are using BitLocker Drive Encryption, changing the Group Policy value to enforce software encryption alone is not sufficient to re-encrypt existing data.

IMPORTANT: You do NOT need to reformat the drive or reinstall any applications after changing BitLocker settings.

...und auch “Lack of Entropy”...

Der zur zuvor genannten Schwachstelle veröffentlichte Forschungsbericht enthält auch einen Abschnitt mit Hinweisen auf die Problematik “Lack of Entropy” zum Boot-Zeitpunkt sowie zu PRNG.

Das Dokument nebst Schwachstellen wurde erstmals veröffentlicht am 5.November 2018.

Auszug:

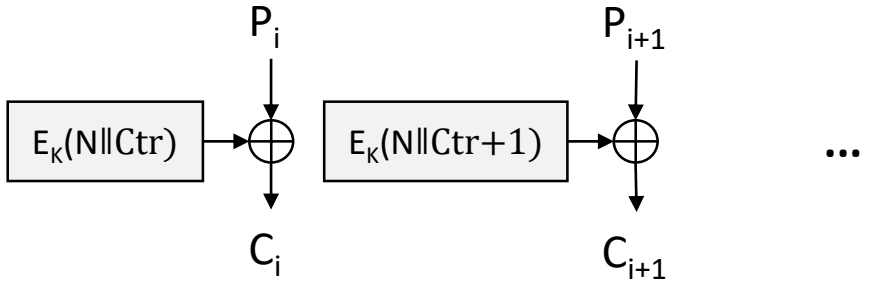
“Lack of entropy in randomly generated DEKs Within the ATA and Opal standards, no means exist for the end user to specify the DEK himself. The only way to affect its value is by randomizing it. This raises the question whether sufficient random entropy is available during the DEK generation. In principle, the environment wherein SSDs are deployed allows for sufficient entropy to be acquired. For example, the drive’s temperature sensor and I/O requests from the host PC. Furthermore, storing and restoring the random pool upon reboots should not be an issue since we are concerned with storage devices. However, random number generators in embedded devices have a notoriously bad reputation.”

CTR Counter Mode (1)

Im Zusammenhang mit Blockchiffren kennen wir einen weiteren Mode. Dieser sogenannte ***Counter Mode*** (kurz: ***CTR***) dient genau genommen der Realisierung einer Stromchiffre, unter Verwendung einer Blockchiffre.

Folgerichtig wird die Blockchiffre auch gar nicht auf den Klartext angewandt, sondern auf hiervon unabhängige Input-Blöcke, mit deren Hilfe ein Schlüsselstrom erzeugt wird, mit dem wir analog zu beispielsweise dem One-Time Pad bitweise per XOR den Klartext verschlüsseln.

Als Input für die Blockchiffre dient eine ***Nonce*** N, in Verbindung mit einem Zähler (Counter) Ctr, der nach jeder Anwendung der Blockchiffre um Eins inkrementiert wird.

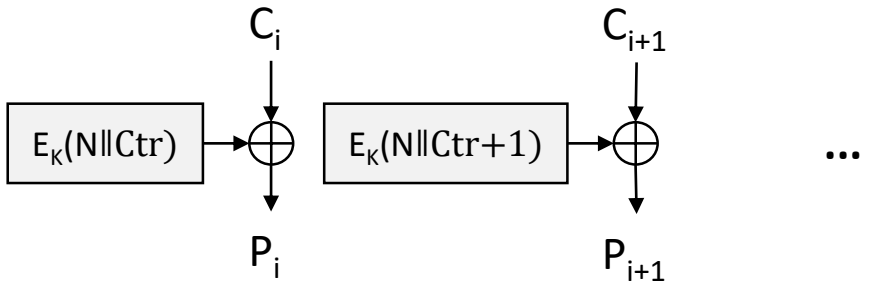


CTR Entschlüsselung

Im Counter Mode gibt es praktisch keinen Unterschied zwischen Verschlüsselung und Entschlüsselung, denn XOR ist invers zu sich selbst (vgl. One-Time Pad).

Dem Kommunikationspartner muss auf vertraulichem Weg nur der Cipher key K übermittelt werden. Die zur Entschlüsselung ebenfalls benötigte Nonce hingegen muss nicht geheimgehalten werden. Der Counter wird bei jedem Block um einen Wert erhöht und ist folglich allen Beteiligten bekannt.

Die Entschlüsselung kann, ebenso wie die Verschlüsselung, an jedem beliebigen Punkt in der Nachricht begonnen werden, da keine Abhängigkeit besteht von vorherigen/benachbarten Message Blocks. Dies erlaubt insbesondere auch eine Parallelisierung von Ver- und Entschlüsselung.



CTR Counter Mode (2)

Die verwendete Nonce N kann jeder beliebige Wert sein und muss keine Anforderungen an kryptographische Schlüssel erfüllen. Sie muss lediglich für jeden Verschlüsselungsvorgang (solange der Cipher key nicht gewechselt wird) anders (einmalig) gewählt werden.

Würde man zweimal denselben Wert wählen für Nonces $N_1 = N_2 = N$, so ergäbe sich (bei gleichem Counter) das gleiche Problem wie bei Wiederverwendung eines Schlüssels beim One-Time Pad:

$C_i = E_K(N \| Ctr) \oplus P_i$ und

$C'_i = E_K(N \| Ctr) \oplus P'_i$

$\Rightarrow C_i \oplus C'_i = (E_K(N \| Ctr) \oplus P_i) \oplus (E_K(N \| Ctr) \oplus P'_i) = P_i \oplus P'_i$

Je nach Beschaffenheit (Entropie) des Plaintexts könnten wir dann aus $P_i \oplus P'_i$ die beiden Plaintexts P_i und P'_i rekonstruieren.

CTR Counter Mode (3)

Die Länge der Nonce sollte nicht zu kurz sein. Beträgt die Länge k Bit, so ist im Durchschnitt nach ca. $2^{k/2}$ Wechseln der Nonces mit einer Wiederholung (d.h. Auftreten einer zuvor bereits verwendeten Nonce) zu rechnen.

Für k = 64 wäre also bereits nach ca. 2^{32} Nonces mit einer Wiederholung zu rechnen, dieser Wert ist also schon relativ klein.

Der Counter Ctr darf sich während eines Verschlüsselungsvorgangs ebenfalls nicht wiederholen, so dass ein hinreichend großes Register vorgesehen werden sollte, beispielsweise 64 Bit. Anderenfalls tritt dasselbe Problem auf wie im Falle wiederholter Nonces (siehe oben).

Andererseits müssen Nonce und Counter beide zusammen in einen Block passen. Angenommen wir haben eine Blocklänge von n = 128. Eine mögliche Aufteilung könnte folgendermaßen sein: 64 Bit für den Counter und 64 Bit für die Nonce. Die Nonce wiederum könnte sich z.B. zusammensetzen aus 48 Bit Message number und 16 weiteren Bits.

CTR Mode ist sehr schnell, nicht nur aufgrund der Parallelisierbarkeit. Vielmehr lässt sich der Schlüsselstrom auch schon vor Beginn der Ver- oder Entschlüsselung berechnen, was im Bedarfsfall einen zusätzlichen Geschwindigkeitsgewinn bedeutet.

OFB Output Feedback Mode (1)

Output Feedback Mode OFB ist ein weiterer Modus, der eine Blockchiffre nutzt, um eine Stromchiffre zu realisieren.

Zunächst erzeugen wir einen Schlüsselstrom, indem wir – ausgehend von einem Initialisierungswert IV, immer wieder die Blockchiffre anwenden auf den vorherigen Block des Schlüsselstroms:

$K_0 = IV$

$K_i = E_K(K_{i-1})$

Die eigentliche Verschlüsselung erfolgt dann wieder in gewohnter Form durch bitweise XOR-Verknüpfung:

$C_i = K_i \oplus P_i$

Die Entschlüsselung ist wieder identisch mit der Verschlüsselung.

OFB Output Feedback Mode (2)

Wird aufgrund z.B. eines Implementierungsfehlers der Initialisierungsvektor mehrfach verwendet, so ist das Verfahren komplett kompromittiert und wir erhalten zwei per XOR übereinanderliegende Klartextströme (ebenso wie bei einem One-time pad mit wiederholtem Key stream). Dies ist in der Praxis ein ernstes Risiko, da solche Fehler absichtlich und unabsichtlich eingebaut werden können.

Der Key stream kann im Voraus berechnet werden. Die Ver- und Entschlüsselung sind dann sehr schnell ausführbar.

OFB erlaubt jedoch nicht, an beliebiger Stelle im Ciphertext mit der Entschlüsselung zu beginnen, wenn nicht zuvor der Key stream vom Anfang der Nachricht beginnend berechnet wurde. Ein möglicher Ausweg wäre die Vorberechnung des gesamten Key streams, wobei aber nur einzelne Blocks innerhalb des Key streams abgespeichert werden, als “Einsprungpunkte”, analog zur Konstruktion von z.B. Rainbow tables.

OFB Output Feedback Mode (3)

Ein Risiko bei Verwendung von OFB besteht darin, dass im Verlauf des Schlüsselstroms zwei erzeugte Key stream Blöcke identisch sind. Rein statistisch ist dies irgendwann der Fall. Bei einer Blocklänge von 128 Bit geschieht dies im Mittel nach etwa 2^{64} erzeugten Schlüsselblöcken (die Ursache liegt auch hier wieder im Birthday paradox).

Tritt dieser Fall ein, so erhalten wir zwei identische Teil-Schlüsselströme. Wieder erhält ein Angreifer sodann zwei per XOR übereinanderliegende Klartextströme.

Hätten wir eine Blocklänge von 64 Bit angenommen, so wäre bereits nach ca. 2^{32} = 4,294,967,296 Blocks mit einer Wiederholung zu rechnen. Hier bewegen wir uns im Gigabyte-Bereich, die Gefahr ist also sehr realistisch.

Ein Ausweg besteht darin, mit einem Cipher key nur eine begrenzte Anzahl Blocks zu verschlüsseln und den Cipher key regelmäßig zu wechseln.

Anmerkung: Szenarien wie oben beschrieben sind die Ursache dafür, dass selbst “ungebrochene” Chiffren evtl. viel weniger “Bits of security” bieten, als dies die Block- und/oder Schlüssellänge zunächst impliziert!

CFB Cipher Feedback Mode (1)

Cipher Feedback Mode CFB ist ein weiterer Betriebsmodus zur Realisierung einer Stromchiffre mithilfe einer Blockchiffre. Die Vorgehensweise ist diesmal etwas anders: bei den bisher betrachteten Verfahren wurden jeweils Output Blöcke erzeugt, deren Größe (Anzahl enthaltener Bits) übereinstimmte mit der Blocklänge der zugrunde liegenden Blockchiffre.

Bei CFB sehen wir, dass dies auch flexibler geht. Input und Output der verwendeten Blockchiffre haben hier zwar dieselbe Größe wie sonst, doch vom Outputblock nutzen wir pro Schritt nicht die komplette Breite, sondern einen Teilabschnitt der Länge m Bits.

Im Inputblock verändern wir jeweils nur einen Teilblock der Länge m und rotieren danach diese m benutzten Bits “heraus”. Dies geschieht mithilfe eines so genannten ***Shift Registers***.

Zur Erläuterung des Verfahrens benötigen wir noch den Begriff des ***Most Significant Bit MSB***. Damit beschreiben wir, ob Datenbits in einem Byte, Register, Block o.ä. “von links nach rechts” oder umgekehrt angeordnet sind (für eine Informatik-fachgerechtere Beschreibung bitte nachlesen). Nachfolgend sei das MSB jeweils das Bit linksaußen im Register bzw. Block.

CFB Cipher Feedback Mode (2)

Ebenso definieren wir das ***Least Significant Bit LSB***. Dies sei für unsere Zwecke das Bit rechtsaußen im Register bzw. Block.

In Analogie hierzu definieren wir zwei Funktionen $\boldsymbol{MSB_m()}$ und $\boldsymbol{LSB_m()}$ auf einem Register R der Länge n, wie folgt:

$MSB_m(R)$ liefert den Teilblock von R, der aus den m Most significant bits besteht.

$LSB_m(R)$ liefert den Teilblock von R, der aus den m Least significant bits besteht.

Im Folgenden seien P_i und C_i Plaintext- bzw. Ciphertext strings der Länge m.

Sei R_i ein Register der Länge n. Dann definieren wir das Register R_{i+1} wie folgt:

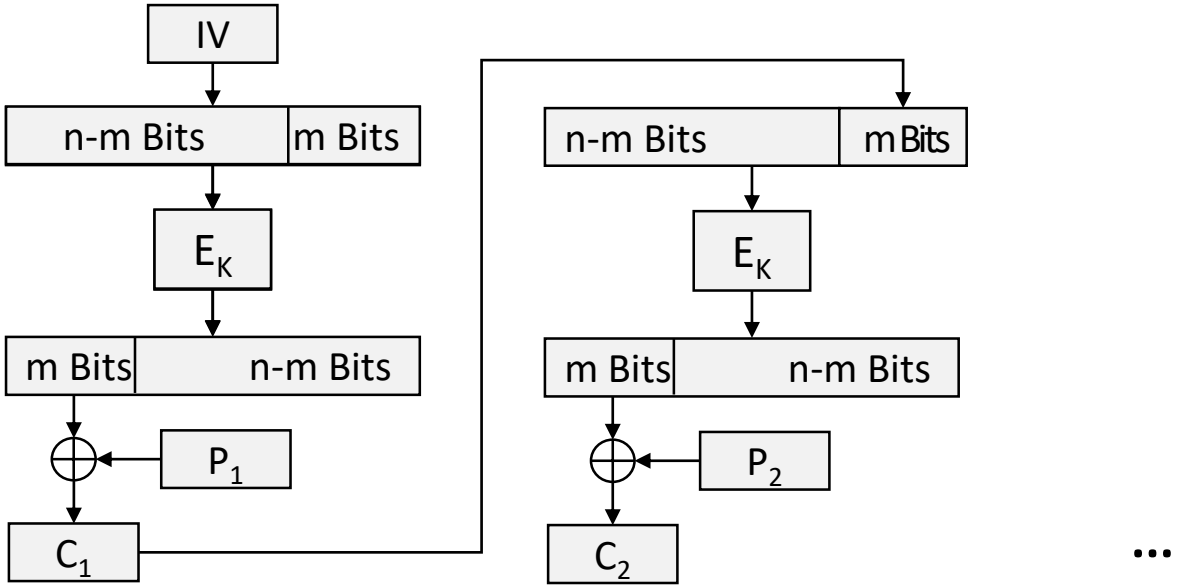
$R_{i+1} = LSB_{n-m}(R_i) \parallel C_i$

Das Register R_i wird also ersetzt durch das Register R_{i+1}, indem aus R_i m Bits nach links herausgeschoben und dafür auf der rechten Seite die m Bits von C_i angefügt werden.

Für R_1 verwenden wir anstelle des nicht existierenden Ciphertexts C_0 einen Initialisierungsvektor: $R_1 = IV$

CFB Cipher Feedback Mode (3)

P_i und C_i haben Blocklänge m, wobei in der Praxis oftmals m = 8. Vom Output der Länge n unserer Blockchiffre nehmen wir nur die m Most significant bits und verknüpfen diese per XOR mit dem Plaintext block P_i. Der resultierende Ciphertext block C_i dient im nächsten Schritt als Input der Blockchiffre. Da C_i nur m Bit liefert, "schieben" wir diese von rechts in unser Schieberegister, wodurch links m Bits herausfallen. Auf diesen Registerinhalt wird erneut die Blockchiffre angewandt, und so fort.



$C_1 = MSB_m(E_K(R_0)) \oplus P_1$
$C_i = MSB_m(E_K(R_i)) \oplus P_i$

wobei R_0 = IV und
$R_{i+1} = LSB_{n-m}(R_i) \| C_i$

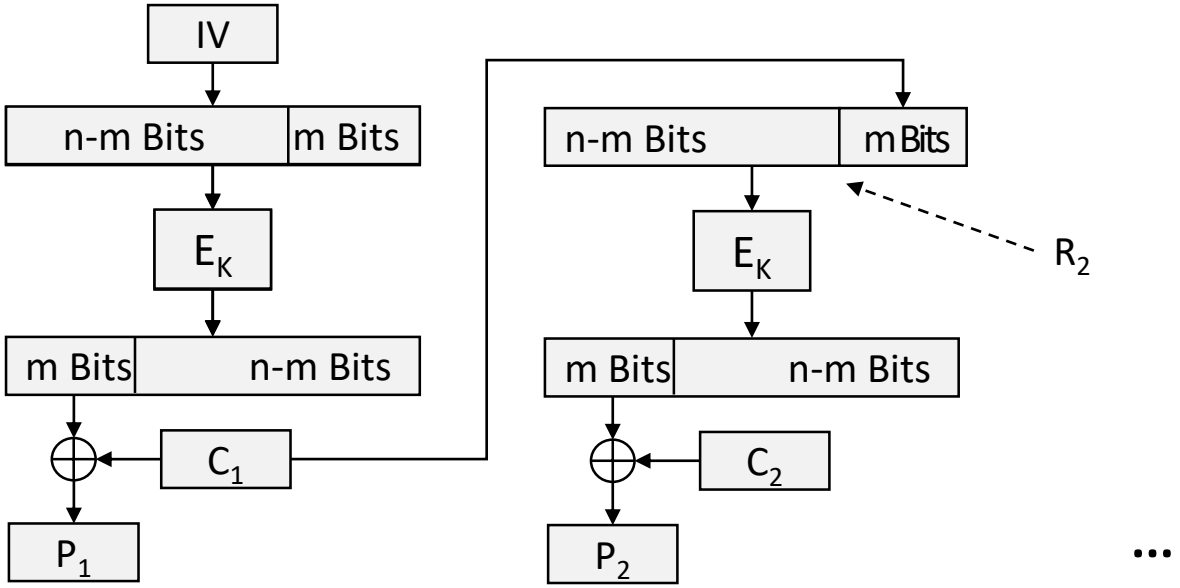
CFB Entschlüsselung

Aus den Gleichungen für die Verschlüsselung erhalten wir unmittelbar:

$P_1 = MSB_m(E_K(R_0)) \oplus C_1 = MSB_m(E_K(IV)) \oplus C_1$

$P_i = MSB_m(E_K(R_i)) \oplus C_i = MSB_m(E_K(LSB_{n-m}(R_{i-1}) \| C_{i-1})) \oplus C_i$

Die Entschlüsselung ist erwartungsgemäß identisch zur Verschlüsselung.



Information Leakage

Ganz gleich wie gut unser Verschlüsselungverfahren ist, erhält ein Beobachter stets gewisse Metadaten wie Kommunikationspartner, Nachrichtengröße etc.

Ist unsere Chiffre “sicher”, so möchten wir allerdings annehmen, dass keine weiteren Informationen über den Plaintext bekannt werden.

Es mag daher überraschen, dass auch bei einer eigentlich sicheren Blockchiffre in Abhängigkeit von den betrachteten Betriebsmodi weitere Informationen über die Nachricht bekannt werden.

Dieses Phänomen bezeichnet man auch als ***Information Leakage*** Problem.

Nachfolgend untersuchen wir einige Beispiele für Information Leakage in den zuvor betrachteten Betriebsmodi.

Information Leakage in CBC (1)

Wir haben am Beispiel ECB gesehen, dass die Kenntnis passender (in diesem Fall: identischer) Chiffretexte genügen kann, um zugehörige identische Klartexte zu rekonstruieren. Dies war auch dann möglich, wenn die zugrunde liegende Blockchiffre keine Schwäche aufweist. Zudem ließen sich einzelne Blocks entfernen, hinzufügen oder in der Reihenfolge vertauschen, ohne dass dies zwingend auffallen muss.

Aus diesem Grund verzichtet man in der Praxis auf ECB und bevorzugt andere Betriebsmodi wie zum Beispiel CBC. CBC verschlüsselt identische Klartexte im Gegensatz zu ECB nicht mehr in identische Chiffretexte. Sind damit alle Probleme gelöst? Es mag zunächst überraschend sein, aber auch CBC hat ein Information Leakage Problem!

Angenommen wir haben einen mittels CBC verschlüsselten Chiffretext. Bei einer hinreichend großen Anzahl Ciphertext blocks besteht die realistische Chance, dass zwei identische Ciphertext blocks C_i und C_j dabei sind.

Schauen wir uns an, was hieraus folgt:

Information Leakage in CBC (2)

Angenommen wir haben zwei identische Ciphertext blocks gefunden:

$C_i = C_j$

$\Leftrightarrow E_K(P_i \oplus C_{i-1}) = E_K(P_j \oplus C_{j-1})$

$\Leftrightarrow P_i \oplus C_{i-1} = P_j \oplus C_{j-1}$

$\Leftrightarrow P_i \oplus P_j = C_{i-1} \oplus C_{j-1}$

Wir müssen also nur die beiden Ciphertext blocks vor C_i und C_j per XOR addieren und erhalten den XOR-verknüpften Klartextblock $P_i \oplus P_j$.

Handelt es sich nicht um sehr "zufällige" Binärdatenblöcke, sondern beispielsweise um Sprachtext, so lassen sich aus $P_i \oplus P_j$ Aussagen über P_i und P_j treffen.

Dies gilt freilich nur für den seltenen Fall, dass zwei Ciphertext Blöcke übereinstimmen.

Information Leakage in CBC (3)

Was gilt für den normalen Fall, dass Ciphertext Blöcke verschieden sind? Für je zwei voneinander verschiedene Ciphertext Blöcke C_i und C_j gilt:

$C_i \neq C_j$

$\Leftrightarrow E_K(P_i \oplus C_{i-1}) \neq E_K(P_j \oplus C_{j-1})$

$\Leftrightarrow P_i \oplus C_{i-1} \neq P_j \oplus C_{j-1}$

$\Leftrightarrow P_i \oplus P_j \neq C_{i-1} \oplus C_{j-1}$

Nehmen wir also an, es sei $C_i \neq C_j$. Für den Fall $P_i = P_j$ resultiert hieraus (s.o.):

$P_i = P_j \Rightarrow 0 = P_i \oplus P_j \neq C_{i-1} \oplus C_{j-1} \Rightarrow C_{i-1} \neq C_{j-1}$

Oder, logisch äquivalent:

$C_{i-1} = C_{j-1} \Rightarrow P_i \neq P_j$

Information Leakage in CTR (1)

Betrachten wir nun, ob auch im Counter Mode Information Leakage auftritt.

Vorab bemerken wir, dass alle Blocks K_i, K_j im Schlüsselstrom des CTR paarweise verschieden sein müssen, da für jeden Block des Key streams ein anderer Input verschlüsselt wird. Da die Blockchiffre eine Permutation auf der Menge ihrer Input Blöcke darstellt, kann jeder Funktionswert $K_i = E_K(Ctr_i)$ nur genau einmal auftreten.

Die Information Leakage bei CTR besteht in unzähligen Ungleichungen der Form

$P_i \oplus P_j \neq C_i \oplus C_j$

Beweis: angenommen, es sei $P_i \oplus P_j = C_i \oplus C_j$ Dann folgt:

$P_i \oplus P_j = C_i \oplus C_j = (P_i \oplus K_i) \oplus (P_j \oplus K_j) = P_i \oplus P_j \oplus K_i \oplus K_j$

$\Rightarrow 0 = K_i \oplus K_j$

$\Rightarrow K_i = K_j$ Dies steht im Widerspruch zur Tatsache $K_i \neq K_j$.

Daher gilt für alle i, j stets: $P_i \oplus P_j \neq C_i \oplus C_j$.

Haben wir eine gewisse Vorkenntnis über die Beschaffenheit des Plaintexts, so können obige Ungleichheitsrelationen wertvolle Hinweise auf den Inhalt geben!

Information Leakage in OFB (1)

Als letztes betrachten wir Information Leakage im Ouput Feeback Mode OFB. Bei OFB wird der Key stream gebildet mittels

$K_i = E_K(K_{i-1})$

Diese Formel erlaubt nicht die Annahme, dass alle Blocks K_i, K_j im Schlüsselstrom des OFB paarweise verschieden sein müssen.

Tritt nun tatsächlich der Fall $K_i = K_j$ auf für zwei i, j mit $i < j$, so wiederholt sich der gesamte Schlüsselstrom zwischen dem Index i und dem Index j ab dem Key stream Block K_j.

Daraus folgt für alle $m > 0$:

$K_{i+m} = K_{j+m} \Rightarrow$

$\Rightarrow C_{i+m} \oplus C_{j+m} = K_{i+m} \oplus P_{i+m} \oplus K_{j+m} \oplus P_{j+m} = P_{i+m} \oplus P_{j+m}$

Wir bekommen also das gleiche Problem wie im Fall eines zweimal verwendeten One-Time Pad Schlüssels.

Die Herausforderung ist in diesem Fall, überhaupt festzustellen, dass obiges Problem aufgetreten ist. Als Angreifer sehen wir nicht die K_i sondern nur die C_i. Je nach Beschaffenheit des Plaintexts lassen sich jedoch statistische Auffälligkeiten identifizieren, sobald zwei XOR-verknüpfte Ciphertext blocks strukturell übereinstimmen mit zwei per XOR übereinander gelegten Plaintext blocks.

Information Leakage in OFB (2)

Das zweite mögliche Information Leakage Problem im OFB tritt dann auf, falls – nicht im Key stream wie im zuvor betrachteten Fall, sondern im Ciphertext stream zwei identische Ciphertext Blöcke $C_i = C_j$ auftreten für zwei $i \neq j$:

$C_i = C_j \Rightarrow$

$\Rightarrow K_i \oplus P_i = K_j \oplus P_j \Rightarrow$

$\Rightarrow P_i \oplus P_j = K_i \oplus K_j$

Dies hilft uns zunächst nicht viel. Sollte jedoch zusätzlich der Fall eintreten, dass hierbei $P_i = P_j$ gelten sollte, so gilt zugleich $K_i = K_j$ und wir erhalten den Hinweis auf Vorliegen des zuvor beschriebenen Information Leakage Problems bei zwei identischen Key stream blocks.

Anmerkung: in vielen Anwendungsfällen ist der Klartextstrom alles andere als zufällig, weist niedrige Entropie auf und verfügt evtl. über viele identische Plaintext blocks. Obiges Szenario ist daher nicht völlig unrealistisch.

Information Leakage in CFB (?)

Betrachten wir wieder analog zu CBC den Fall, dass wir zwei identische Ciphertext blocks gefunden haben. Dies bedeutet:

$C_i = C_j$

$\Leftrightarrow MSB_m(E_K(LSB_{n-m}(R_{i-1}) \| C_{i-1})) \oplus P_i = MSB_m(E_K(LSB_{n-m}(R_{j-1}) \| C_{j-1})) \oplus P_j$

$\Leftrightarrow MSB_m(E_K(LSB_{n-m}(R_{i-1}) \| C_{i-1})) \oplus MSB_m(E_K(LSB_{n-m}(R_{j-1}) \| C_{j-1})) = P_i \oplus P_j$

$P_i \oplus P_j$ ist also eine XOR-Verknüpfung der Outputs zweier Blockverschlüsselungen. Hieraus lässt sich bei "sicherer" Blockchiffre nichts sinnvolles ableiten für $P_i \oplus P_j$.

Der Fall $C_i \neq C_j$ führt in dieselbe Sackgasse.

Identische Plaintexts werden offenkundig auf verschiedene Ciphertexts abgebildet. Insgesamt verhält sich CFB also zufriedenstellend bezüglich Information Leakage.