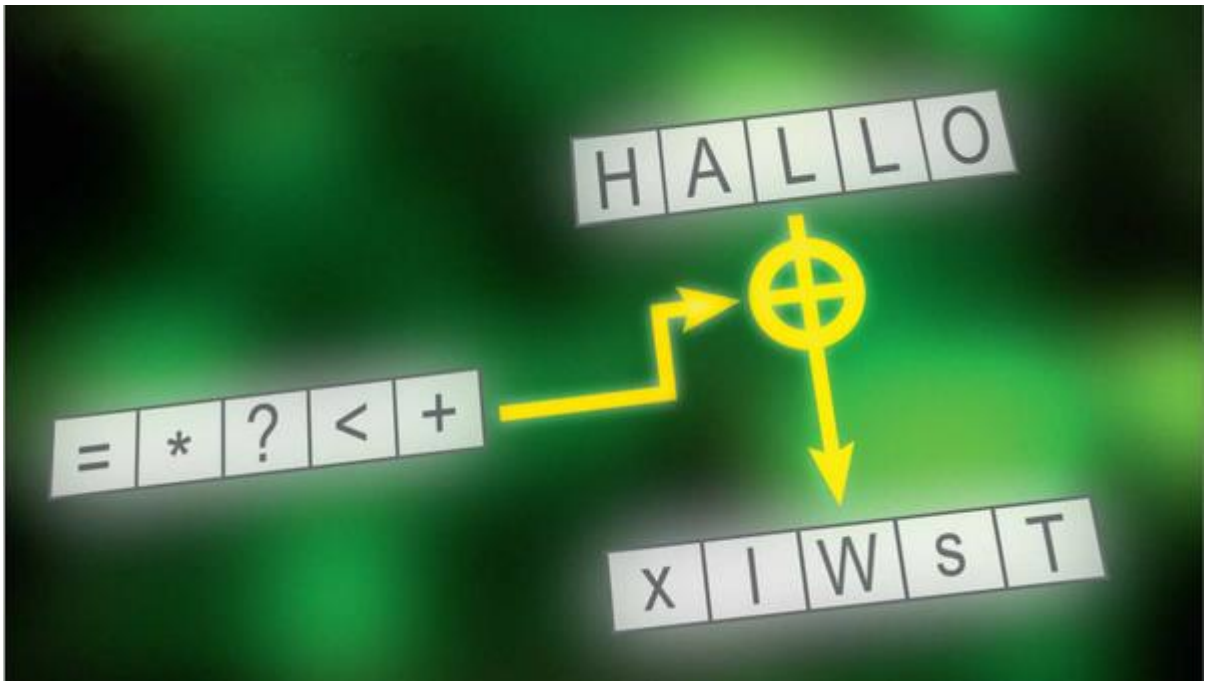
Kryptographie in der IT - Empfehlungen zu Verschlüsselung und Verfahren

Dieser Artikel erschien ursprünglich in c't 01/2016, Seite 174



Kryptographie ist ein wichtiger Baustein moderner IT – Sicherheit, Vertraulichkeit und Privatsphäre hängen davon ab. Der folgende Krypto-Wegweiser gibt einen kompakten Überblick zu den aktuell relevanten Verfahren.

Jürgen Schmidt - *17.06.2016*

Dieser Krypto-Wegweiser wird Sie nicht zum Krypto-Experten machen, der selbst sichere Systeme entwerfen und realisieren kann. Dazu ist die Materie dann doch zu komplex. Aber wenn Sie einen kompakten Überblick zum Stand der Dinge möchten, etwa um die Krypto-Funktionen einer Anwendung oder eines Dienstes zu beurteilen, bekommen Sie das nötige Hintergrundwissen in verständlicher und kompakter Form.

Der Wegweiser konzentriert sich dazu auf die elementaren Grundbausteine, aus denen sich moderne Krypto-Systeme zusammensetzen. Er erklärt, was wozu verwendet wird, welche der Verfahren sicher sind, wo mit Problemen zu rechnen ist und wo die wichtigen Baustellen für die absehbare Zukunft liegen. Bevor es richtig losgeht, jedoch ein kurzer Ausflug dazu, wie man Sicherheit messbar macht. Ungeduldige können direkt zu den Hashes und MACs springen.

Sicherheit der Verschlüsselung

Bereits 1883 formulierte Auguste Kerckhoff den Grundsatz moderner Kryptographie. Demnach beruht die Sicherheit von Verschlüsselung nicht auf der Geheimhaltung des Verfahrens, sondern auf der Geheimhaltung der Schlüssel. Wer mit „streng geheimer“, „patentierter“ Verschlüsselung wirbt, versucht Ihnen „Schlangenöl“ zu verkaufen. Gegen motivierte Angreifer hilft die in etwa so gut wie dieses Allheilmittel der Wunderheiler des Wilden Westens. Gute Verschlüsselung setzt auf bekannte und gut analysierte Verfahren.

Die Sicherheit wächst dabei mit der Länge und somit der Zahl der möglichen Schlüssel. 256-Bit-Schlüssel bedeuten 2^{256} mögliche Variationen. Wenn das Verfahren keine Schwäche aufweist, muss man statistisch gesehen die Hälfte davon durchprobieren, um an den Klartext zu gelangen.

Noch längere Schlüssel wären Verschwendung, da sich schon 2^{255} Schlüssel nicht mehr durchprobieren lassen, ohne grundsätzliche physikalische Barrieren zu überwinden. Um es mit Bruce Schneier zu sagen: Brute-Force-Angriffe gegen 256-Bit-Schlüssel werden solange unmöglich bleiben, wie sich die dazu eingesetzten Computer aus Materie zusammensetzen und Raum einnehmen ("until computers are built from something other than matter and occupy something other than space").

Um Sicherheit vergleichbar zu machen, schätzt man die Zahl der benötigten Operationen, einen Code ohne den Schlüssel zu dechiffrieren. Als Referenz benutzt man ein symmetrisches Verfahren wie AES, dem mit 128-Bit-Schlüsseln auch 128-Bit-Sicherheit zugeschrieben wird. Das Knacken von RSA-Verschlüsselung erfolgt über das Zerlegen großer Zahlen in ihre Primfaktoren, wofür es Optimierungsverfahren wie das Zahlkörpersieb (Number Field Sieve) gibt. Damit bleibt für 1024-Bit-RSA lediglich eine Sicherheit von 80 Bit übrig. Weitere Optimierungen, die diesen Wert weiter reduzieren, können jederzeit auftauchen.

Hashes und MACs

Kurz: Nimm SHA-256.

Ein zentraler Krypto-Baustein sind kryptografische Hash-Funktionen. Sie kommen zum Einsatz, um sicherzustellen, dass Daten etwa bei einer Übertragung nicht verändert wurden; gelegentlich finden sich auch Hash-Werte als Kontrollangabe bei Download-Angeboten. Außerdem speichern moderne Systeme statt Passwörtern nur deren Hashes.

Eine Hash-Funktion erzeugt aus einem beliebig großen Datensatz eine Art Fingerabdruck von beispielsweise 256 Bit, der als hochsichere Prüfsumme fungieren kann. Denn jede noch so kleine Änderung an den Daten ändert den Hash-Wert. Man kann außerdem für einen vorgegebenen Hash-Wert (mit endlichem Aufwand) keinen Datensatz finden, der diesen Hash-Wert ergibt.

Noch schwerer zu erfüllen, aber wünschenswert ist die Kollisionsresistenz: Es ist praktisch nicht durchführbar, zwei Nachrichten zu finden, die den gleichen Hash-Wert erzeugen. Diese Eigenschaften sind erforderlich, damit ein Hash-Wert nicht nur gegen versehentliche Veränderungen, sondern auch vor absichtlichen Fälschungen schützt.

Konkret nutzt man häufig Message Authentication Codes (MACs beziehungsweise Hashed MACs, HMACs), bei denen zusätzlich zu den Daten noch ein Geheimnis in den Hash-Vorgang mit eingeht. Dieses Geheimnis kennen nur Absender und Empfänger. Stimmt beim Check des Empfängers der selbst berechnete MAC mit dem mitgeschickten MAC überein, stammen die Daten tatsächlich vom erwünschten Absender und wurden unterwegs nicht verändert.

Empfehlungen zu Hash-Funktionen

MD5, lange Zeit eine der am weitesten verbreitete Hash-Funktion, ist geknackt und sollte nicht mehr zum Einsatz kommen. **SHA-1** ist wegen mangelhafter Kollisionsresistenz ebenfalls problematisch und sollte insbesondere bei neuen Projekten vermieden werden. Dafür hat sich das durch die erfolgreichen Angriffe auf SHA-1 angeschlagene Vertrauen in dessen eng verwandten Nachfolger **SHA-2** gefestigt. Dessen Varianten **SHA-256**, **SHA-384** und **SHA-512** gelten als sicher.

Sollten sich doch Schwächen in SHA-2 finden, steht mit **SHA-3** aka **Keccak** auch bereits ein Nachfolger mit einem komplett anderen Aufbau bereit. Die Hauptkritik an SHA-3 gilt dem recht hohen Ressourcenbedarf; in diesem Bereich punktet **Blake2**, dessen Vorgänger erst im Finale des SHA-3-Shootouts gegen Keccak den Kürzeren zog. Ein Exot ist der ebenfalls standardisierte MAC **Poly1305**; er setzt statt einer Hash-Funktion das symmetrische Verschlüsselungsverfahren AES ein, um seine Prüfcodes zu erstellen.

Für das Speichern von Passwörtern sollte man keine Hash-Funktionen direkt verwenden, sondern stattdessen Verfahren wie **PBKDF2**, **bcrypt** oder **scrypt** einsetzen. Die sind absichtlich so konzipiert, dass ein Passwort-Test möglichst viel Ressourcen benötigt. Sie erreichen dies unter anderem, indem sie viele Iterationen über Hash-Operationen durchführen. Auch für das Erzeugen von geheimen (AES-) Schlüsseln aus Passwörtern sind diese PBKDF2, bcrypt und scrypt das Mittel der Wahl.

Fazit: Bei den Hash-Funktionen und MACs ist alles im grünen Bereich. Es sind ausreichend Reserven vorhanden, um auch auf Überraschungen angemessen zu reagieren.

Symmetrische Verschlüsselung

Kurz: Nimm AES mit 256 Bit.

Die eigentliche Verschlüsselung von Daten erfolgt in aller Regel mit einem symmetrischen Verfahren – das ist schnell und einfach. Symmetrisch heißt es deshalb, weil zum Ver- und Entschlüsseln das gleiche Geheimnis zum Einsatz kommt. Das müssen also sowohl Absender als auch Empfänger kennen – und tunlichst nur die.

Der unangefochtene Platzhirsch in diesem Bereich ist der Advanced Encryption Standard (**AES**). Es sind keine nennenswerten Schwächen bekannt; so gut wie alle angesehenen Kryptologen vertrauen ihm. Als gute Alternative gilt das unter anderem von der europäischen ENISA empfohlene **Camellia**, das allerdings deutlich langsamer ist. Beides sind sogenannte **Block-Verschlüsselungsverfahren**, die immer mit Datenblöcken fester Größe arbeiten.

Stromverschlüsselung arbeitet hingegen einen kontinuierlichen Datenstrom Byte für Byte ab. Eine solche Stream Cipher bietet Vorteile, etwa wenn man einzelne Daten im verschlüsselten Datenstrom exakt lokalisieren will. Das am weitesten verbreitete **RC4** gilt als gebrochen oder zumindest sehr angeschlagen; man sollte es nicht mehr einsetzen. Schlimmer noch sind die beim Mobilfunk oft eingesetzten A5/1 und A5/2 – die kann man bereits in Echtzeit knacken.

Die vielversprechendste Stream Cipher ist **ChaCha** von Dan J. Bernstein; sie wird vor allem von Google in der Kombination ChaCha20-Poly1305 für HTTPS verwendet. Allerdings ist die Standardisierung für TLS 1.2 noch nicht abgeschlossen. Als Schlüssellänge reichen derzeit 128 Bit aus; für langfristige Sicherheit sollte man 256- Bit-Schlüssel einsetzen.

Fazit: Auch bei den symmetrischen Verfahren ist alles okay; Probleme sind nicht in Sicht und für den Fall, dass doch welche auftauchen, gibt es ausreichend Reserven und Alternativen.

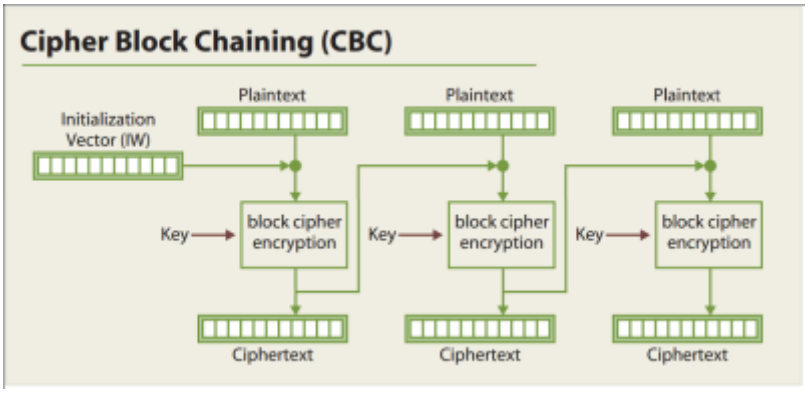
Die Bilddaten des c't-Logos links wurden zweimal mit einem jeweils 16-stelligen Passwort AES-verschlüsselt. Während die Verschlüsselung mit Cipher Block Chaining (CBC) nur Rauschen ohne sichtbare Strukturen zeigt (Mitte), ist im simplen Electronic Codebook Modus (ECB) die Form noch deutlich zu erkennen (rechts).

Betriebsmodi von CBC bis GCM

Kurz: Nimm AES-GCM.

Wenn man mit einer Block-Chiffre wie AES mehr als 32 Byte (256 Bit) verschlüsseln möchte, muss man sich überlegen, mit welchem Schlüssel man den zweiten und die folgenden Blocks chiffriert. Das legt der Betriebsmodus der Verschlüsselung fest, auch Cipher Mode genannt.

Immer wieder den gleichen Schlüssel zu nehmen (Electronic Codebook Modus, kurz **ECB**), verbietet sich, da damit gleiche Daten immer den gleichen Chiffretext erzeugen. Daraus können Angreifer oft schon sehr viel ableiten. Der am häufigsten genutzte Modus ist das Cipher Block Chaining (**CBC**), bei dem der Cipher-Text des jeweils vorgehenden Blocks in die Verschlüsselung mit eingeht. Dieses Verfahren weist jedoch Schwächen auf, die auch immer wieder zu realen Angriffen wie POODLE führten.



Beim Cipher Block Chaining geht der Ciphertext eines Blocks in die Verschlüsselung des nächsten mit ein.

Unter anderem müssen Datensätze immer auf ein Vielfaches der vollen Blockgröße verlängert werden. Auf Grund einer schlechten Entscheidung beim Design von SSL sind diese Fülldaten nicht integritätsgeschützt und lassen sich somit zum Durchprobieren bestimmter Entschlüsselungsmöglichkeiten missbrauchen. Ein solches „Padding Oracle“ war die Basis des spektakulären POODLE-Angriffs auf HTTPS-Verschlüsselung. Darüber hinaus lässt sich die Verschlüsselung großer Datenmengen mit CBC nicht parallelisieren.

Die bekannten Angriffe auf Verschlüsselung im CBC-Modus wurden immer recht schnell gefixt, sodass man es nach wie vor benutzen kann. Doch eigentlich will man weg davon. Die beste Alternative ist der Galois Counter Mode (**GCM**), der mit fortlaufenden Zählern arbeitet und voll parallelisierbar ist. Die wichtigste Eigenschaft ist jedoch, dass GCM die Integritätssicherung via MAC mit der Verschlüsselung kombiniert.

Diese Authenticated Encryption (**AE**) beziehungsweise auch Authenticated Encryption with Associated Data (**AEAD**) korrigiert endlich die vor vielen Jahren für SSL getroffene Fehlentscheidung,

zuerst den MAC über die Daten zu bilden und dann erst zu verschlüsseln (MAC-then-Encrypt, siehe Padding Oracle). Für TLS 1.2 sind AES-GCM und Camellia-GCM standardisiert. Google setzt stattdessen auf die Strom-Chiffre ChaCha20 mit dem MAC Poly1305.

Speziell für die Verschlüsselung von Festplatten kommt neben CBC oft **AES-XTS** zum Einsatz. Das splittet den kompletten Schlüssel in zwei Hälften auf. Für eine echte AES-Verschlüsselung mit 256 Bit muss man also XTS mit 512-Bit-Schlüsseln füttern.

Fazit: Der aktuelle Stand der Dinge ist nicht wirklich schön, aber es ist Besserung in Sicht. Kein Grund zur Panik.

Asymmetrische Verschlüsselung

Kurz: Nutze ECC-Verfahren.

Asymmetrische Verschlüsselung kennen die meisten von verschlüsselter E-Mail-Kommunikation mit PGP; im Hintergrund verwenden sie aber fast alle moderne Krypto-Systeme. Dabei kommen zwei Schlüssel zum Einsatz: ein öffentlicher, den man frei verteilen kann, und ein geheimer, den man unter Verschluss halten muss. Die beiden ergänzen sich derart, dass man die Daten, die mit dem öffentlichen Schlüssel verschlüsselt wurden, nur mit dem geheimen wieder dechiffrieren kann – und umgekehrt.

Asymmetrische Verschlüsselung ist viel langsamer als die symmetrische und kommt deshalb vor allem für Dinge zum Einsatz, wo nur kleine Datenmengen anfallen. Bei digitalen Signaturen sichert man einen Hash-Wert über die zu signierenden Daten mit seinem geheimen Schlüssel. Beim Schlüsselaustausch tauscht man mit seinem Gegenüber auf sicherem Weg einen geheimen (AES-)Schlüssel für die anschließende symmetrische Verschlüsselung der eigentlichen Nutzdaten aus.

Das prominenteste und am häufigsten genutzte asymmetrische Verfahren ist **RSA**, das sowohl für digitale Signaturen als auch für den Schlüsselaustausch zum Einsatz kommt. Bei letzterem hat ihm aber mittlerweile Diffie Hellman (**DH**) den Rang abgelaufen. Bei DH gehen in jeden einzelnen Schlüsselaustausch Zufallszahlen ein, die die Kommunikationspartner anschließend verwerfen. Somit kann ein Angreifer – anders als bei RSA – den geheimen AES-Schlüssel zu einem späteren Zeitpunkt nicht mehr aus den aufgezeichneten Chiffredaten rekonstruieren (Forward Secrecy). Für digitale Signaturen kommt vereinzelt **DSA** zum Einsatz.

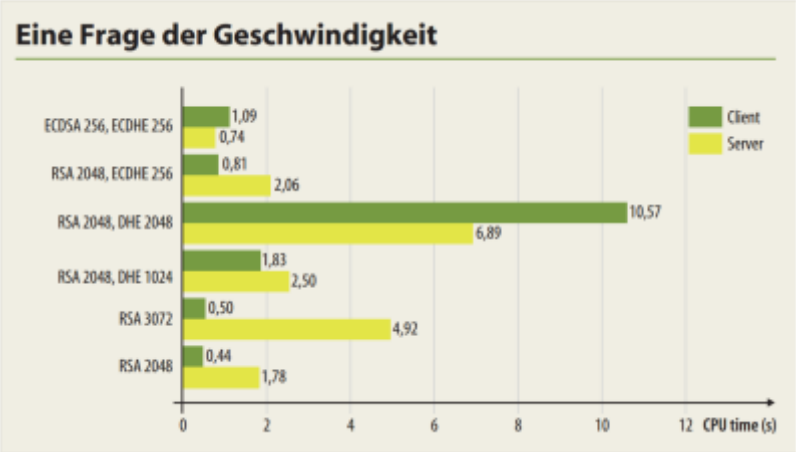
Asymmetrische Verfahren beruhen darauf, dass sich bestimmte mathematische Probleme praktisch nicht mehr lösen lassen, wenn die involvierten Zahlen sehr groß sind. So setzt RSA darauf, dass man zwar sehr schnell Multiplizieren, aber ausreichend große Zahlen nicht mehr in endlicher Zeit in ihre PrimFaktoren zerlegen kann. Analog setzen DH und DSA auf die Asymmetrie zwischen dem schnellen Potenzieren und dessen Umkehrung – dem aufwendigen Berechnen des Logarithmus beim Rechnen mit Teilresten.

Doch die kontinuierlich wachsende Rechenleistung aktueller Computer knackt diese Aufgaben immer schneller. Dem muss man durch immer längere Schlüssel Rechnung tragen. Besonderes besorgniserregend ist, dass die benötigte Schlüssellänge nicht linear anwächst. So entsprechen 1024-Bit-Schlüssel für RSA und DH gerade mal der Sicherheit eines symmetrischen Verfahrens mit 80 Bit.

Das ist durchaus schon innerhalb der Reichweite eines motivierten Angreifers; tatsächlich wurde bei der Faktorisierung der Mersenne-Zahl 2^{1039}-1 bereits ein Co-Teiler mit 1017 Binärstellen errechnet. Das Knacken von Diffie Hellman mit 1024 Bit ist grob geschätzt mit Hardware für etwa 100 Millionen

US-Dollar möglich. Das klingt viel – ist mit dem Budget der NSA aber durchaus bereits realistisch. Und die Hardware wird immer schneller.

Eine Verdoppelung der Schlüssellänge auf 2048 Bit ergibt nicht etwa 160, sondern lediglich 112-Bit-Sicherheit. Das ist in etwa das Sicherheitsniveau von Triple-DES und reicht gemäß ENISA-Richtlinien gerade mal für kurzfristige Sicherheit.



Die Dauer des TLS-Verbindungsaufbaus steigt dramatisch mit der Schlüssellänge für Signatur und Schlüsselaustausch (Angaben für 1000 Handshakes). Erst durch den Einsatz von EC-Verfahren erreicht die Belastung wieder ein akzeptables Maß.

Bild: Ivan Ristic, Bulletproof SSL and TLS

Empfehlungen für asymmetrische Verschlüsselung

Schon für einen mittelfristigen Sicherheitsbedarf empfiehlt die EU-Sicherheitsbehörde das Äquivalent zu 128 Bit, was bei RSA in etwa 3072-Bit-Schlüssel bedeutet. Für langfristig sicher aufzubewahrende Daten müsste man demnach schon heute Schlüssel mit 15360 Bit nutzen. Das macht natürlich niemand, denn solch lange Schlüssel bedeuten ernsthafte Performance-Probleme.

Ganz grob führt eine Verdoppelung der Schlüssellänge zu sechs bis sieben Mal längeren Entschlüsselungszeiten. Da bedeuten 2048 Bit auf Web-Servern und schwachbrüstigen Embedded Devices schon jetzt ein Problem.

Fazit: Das Minimum für sicheres RSA und DH sind 2048 Bit; neue PGP-Schlüssel sollte man mit 4096 Bit erzeugen. Schon mittelfristig brauchen wir Alternativen.

Elliptische Kurven Verschlüsselung

Kurz: Ja, aber …

Die beste Alternative zu herkömmlicher, asymmetrischer Verschlüsselung sind derzeit Krypto-Systeme auf elliptischen Kurven. Diffie Hellman, RSA und DSA arbeiten mit ganzen Zahlen. Elliptic Curve Cryptography (ECC) benutzt stattdessen Punkte auf elliptischen Kurven. Mathematiker haben für diese dann Operationen wie die Addition und Multiplikation definiert (sie nennen das dann einen Körper).

Durch mehrfache Multiplikation hat man das Potenzieren und dessen Umkehrung – den diskreten Logarithmus. Und damit kann man alle Verfahren, die auf dem Problem der Berechnung diskreter Logarithmen beruhen, auf elliptische Kurven übertragen. Damit kann man **ECDSA** für digitale

Signaturen und **ECDH** für den Schlüsselaustausch benutzen. Beides ist bereits standardisiert; alle halbwegs aktuellen TLS-Implementierungen auf Client- und Server-Seite unterstützen ECC.

Der Hauptvorteil: Da die Rechenoperationen auf elliptischen Kurven deutlich aufwendiger sind und sich auf Computern schlechter optimieren lassen als bei ganzen Zahlen, kann man wieder mit kürzeren Schlüsseln arbeiten. Das entlastet nicht nur Server, sondern kommt insbesondere den Entwicklern von Applikationen auf Embedded-Geräten etwa für das Internet of Things entgegen.

Konkret bieten bereits ECC-Schlüssel mit 256 Bit ein Sicherheitsniveau vergleichbar zu AES mit 128 Bit. Und das Schöne daran: Eine einfache Verdoppelung auf ECC-Schlüssel mit 512 Bit genügt bereits, um auch die langfristige 256-Bit-Sicherheit zu realisieren.

Der Pferdefuß bei dem Ganzen ist, dass bisher nur die von der NIST spezifizierten Kurven große Verbreitung gefunden haben. Da bei der Wahl der Parameter die NSA die Hand im Spiel hatte, standen die jedoch zunächst unter dem Verdacht, der US-Geheimdienst könnte sich einmal mehr eine Hintertür eingebaut haben. Diese Besorgnis hat sich in der Krypto-Szene weitgehend gelegt.

Auch ausgewiesene NSA-Kritiker wie Matthew Green und Bruce Schneier halten eine ECC-Hintertür der NIST-Kurven mittlerweile für nahezu ausgeschlossen. Doch es gibt andere Kritik. So sind die Kurven nicht optimal, was die Performance angeht, und die konkrete Implementierung in Krypto-Systemen ist kompliziert und damit fehleranfällig. Im Prinzip sind sich alle Experten einig, dass man eigentlich ECC auf Basis der von Dan J. Bernstein vorgeschlagenen **Curve25519** haben möchte. Das setzen bereits OpenSSH, Apple für iOS und einige weitere Projekte erfolgreich ein.

Doch die Standardisierung von Curve25519 für TLS zieht sich in einem Trauerspiel der konsensorientierten Gremienarbeit seit Jahren hin. Mittlerweile ist Curve25519 in RFC 7748 standardisiert. Für die Nutzung im Rahmen von TLS ist draft-ietf-tls-rfc4492bis-07 zumindest auf der Zielgeraden; Googles Chrome unterstützt es bereits.

Curve25519 bietet ein Security-Level von etwa 128 Bit, was allgemein als ausreichend gilt. Für höhere Ansprüche spezifiziert RFC 7748 auch die sogenannte **Goldilocks**-Kurve (Curve448) mit 224 Bit, die aber bei weitem nicht so ausgereift ist. Als Alternative zu den NIST-Kurven vor allem im Bereich der Embedded-Geräte gibt es noch die ebenfalls schon standardisierten **Brainpool**-Kurven. Außerhalb des Einflussbereichs des BSI haben die jedoch wenig Fürsprecher gefunden.

Fazit: Man will ECC eigentlich schon heute; am besten mit Curve25519. Doch das Gerangel um die Standards von morgen ist immer noch in vollem Gang.

Was kommt danach?

Kurz: Wir brauchen mehr Post Quantum Crypto.

Während sich die Beteiligten noch um die richtigen ECC-Standards balgen, hängt über ihnen schon das Damokles-Schwert der Quanten-Computer. Auf denen lässt sich nämlich Shors Algorithmus zum Berechnen von Primfaktoren und Logarithmen so effizient umsetzen, dass sowohl RSA, Diffie Hellman und DSA als auch ihre ECC-Äquivalente ECDSA und ECDH als geknackt gelten müssen. Symmetrische Verschlüsselung, etwa mit AES, ist davon nicht betroffen.

Google und die NASA haben sich bereits einen QuantenComputer von D-Wave mit 512 Qubits angeschafft.

Bild: D-Wave Systems

Um es noch einmal zu betonen: Quanten-Computer bedeuten das Ende aller derzeit etablierten Public-Key-Verfahren unter anderem für digitale Signaturen und Schlüsselaustausch. Damit bricht ein beträchtlicher Teil des Fundaments aktueller Krypto-Systeme komplett weg. Adäquater Ersatz ist bislang nicht in Sicht.

Die öffentliche Forschung im Bereich Quanten-Computer steckt noch in den Kinderschuhen; die größte mit Shors Verfahren bisher faktorisierte Zahl ist 21. Mit einem auf Quanten-Computern umgesetzten Verfahren zum Auffinden eines globalen Minimums gelang es immerhin schon 56 153 zu faktorisieren.

Das mag man belächeln, ist es doch um viele Größenordnungen von den Möglichkeiten klassischer Computer entfernt. Doch niemand weiß, was die NSA bereits in ihren Kellern stehen hat. Sicher ist nur, dass sie einen beträchtlichen Teil ihres Milliarden-Budgets in diese Rechenmaschinen der nächsten Generation investiert.

Post-Quantum-Kryptographie

Post Quantum Cryptography (**PQC**) ist folglich eines der wichtigsten aktuellen Forschungsgebiete. So fördert die EU das Forschungsprojekt PQCRYPTO; das BSI experimentiert bereits mit IPSec-Implementierungen, die Diffie-Hellman-Schlüsselaustausch durch ein **Niederreiter**-Kryptosystem ersetzen. Die NSA hat sogar einen konkreten Leitfaden und Zeitplan für den Umstieg auf PostQuantum-Crypto angekündigt.

Von der realen Praxis ist das alles jedoch noch sehr weit weg. Angesichts dessen, wie viele Jahre es dauert, kaputte Verfahren wie MD5, SHA1 und RSA-1024 auszumustern oder bereits praxiserprobte Verfahren wie Curve25519 zu standardisieren und einzuführen, kann man schon langsam nervös werden. Zwar sind auch die Quanten-Computer, die 2048-Bit-Zahlen faktorisieren, noch nicht gebaut – jedenfalls hoffen wir das. Doch die Uhr tickt.

Fazit: Post-Quantum-Crypto muss einsatzbereit sein, bevor die ersten echten Quanten-Computer ihren Betrieb aufnehmen. Dazu ist noch einiges an Forschung, Standardisierung und Tests erforderlich. (ju)

URL dieses Artikels:

http://www.heise.de/security/artikel/Kryptographie-in-der-IT-Empfehlungen-zu-Verschluesselung-und-Verfahren-3221002.html

Links in diesem Artikel:

[1] #Hashes

[2] https://www.rfc-editor.org/rfc/rfc7748.txt

[3] https://tools.ietf.org/html/draft-ietf-tls-rfc4492bis-07

[4] http://pqcrypto.eu.org/

[5] http://www.heise.de/meldung/NSA-bereitet-Post-Quanten-Kryptographie-vor-2789403.html