رمزنگاری

Basic concepts of computer cryptography

www.elnazmohamadi.com

فهرست مطالب

۴-۱. مقدمه‌ای بر رمزنگاری

کلمهٔ "Cryptography" از زبان یونانی گرفته شده‌است و وقتی‌که واژه به واژه (تحت‌اللفظی) ترجمه شود، "نوشتن محرمانه" معنی می‌دهد. قبل از ظهور ارتباطات دیجیتالی، رمزنگاری اصولاً بوسیلهٔ ارتش برای اهداف جاسوسی استفاده می‌شد. با پیشرفت در ارتباطات نوین، تکنولوژی، شرکتها و افراد را به نقل و انتقالات اطلاعات با هزینه‌ای بسیار پایین از طریق شبکه‌های همگانی نظیر اینترنت قادرکرده است. این ترقی در عوض هزینه‌ای معادل امکان افشاء داده‌های انتقال‌یافته از طریق چنین واسطه‌ای را دربر دارد. بنابراین ضروری است که مطمئن شویم داده‌های حساس به روشی غیرقابل نفوذ و امن در شبکه‌های همگانی از نقطه‌ای به نقطه‌ای دیگر منتقل می‌شوند. رمزنگاری به ما کمک می‌کند که با غیرمفهوم و پیچیده‌کردن پیامها برای همه بجز گیرندهٔ دلخواه به این هدف دست پیدا کنیم.

هر قدر که در یک جامعهٔ اطلاعاتی پیش می‌رویم، ابزار فنی برای نظارت سراسری میلیونها فرد برای دولت‌های بزرگ امکان‌پذیر شده‌است. درواقع رمزنگاری، به یکی از ابزارهای اصلی برای پوشیدگی، اطمینان، کنترل دسترسی، پرداختهای الکترونیکی، امنیت شرکتی و بسیاری از زمینه‌های دیگر تبدیل شده است. درادامه، واژگان پایه (اصطلاحات علمی یا فنی) و روشهای اصلی رمزنگاری بیان خواهند شد.

۴-۱-۱. اصطلاحات علمی پایه[1]

فرض کنید که شخصی می‌خواهد یک پیام را به شخص دیگری که دریافت‌کنندهٔ پیام خواهد بود، بفرستد و می‌خواهد مطمئن باشد که هیچکس دیگری نمی‌تواند پیام را بخواند. در هر حال، این احتمال وجود دارد که شخص دیگری نامه را بازکند یا مکاتبه(ارتباط) الکترونیکی را بشنود.

در اصطلاح رمزنگاری، پیام‌اصلی plaintext یا cleartext نامیده می‌شود. رمزگذاری محتویات پیام به نحوی که محتوای آن را از بیگانگان مخفی کند، پنهان‌کردن (Encryption) نامیده می‌شود. پیام‌پنهان‌شده (رمزشده) ciphertext نامیده می‌شود. به فرآیند بازیابی plaintext از ciphertext، آشکارسازی[2] گفته می‌شود. در فرآیندهای پنهان سازی و آشکار سازی به طور معمول از کلید استفاده می‌شود و روش رمزنگاری به‌گونه‌ای است که آشکارسازی تنها با دانستن کلید مناسب می‌تواند انجام شود.

رمزنگاری[3] هنر یا علم محرمانه نگاهداشتن پیامها است. شکستن رمز[4] هنر شکستن[1] رمز کننده‌ها می‌باشد؛ بدین معنی که plaintext بدون دانستن کلید مناسب بازیابی شود. افرادی که رمزنگاری را انجام می‌دهند، رمزنگاران[2] هستند، وشاغلان cryptanalysis، cryptanalystها می‌باشند.

[1] Basic Terminology
[2] Decryption
[3] Cryptography
[4] Cryptanalysis

رمز نگاری با تمام جنبه‌های پیغام‌رسانی امن، تصدیق[3]، امضاهای دیجیتالی، پول الکترونیکی ودیگر کاربردها سر وکار دارد. Cryptology یک شاخه از ریاضیات است که پایه‌های ریاضی روشهای پنهان سازی (رمز نگاری) را مطالعه و بررسی می کند.

۴-۱-۲. طبقه‌بندی الگوریتم‌های رمزنگاری

یک روش encryption و decryption، رمزنگار[4] نامیده می‌شود. بعضی روشهای پنهان‌سازی بر محرمانه‌بودن الگوریتم‌ها متکی هستند، چنین الگوریتم‌هایی تنها از بعد تاریخی اهمیت دارند و برای نیازهای جهان واقعی کافی نیستند. همهٔ الگوریتم‌های مدرن برای کنترل encryption و decryption ازکلید استفاده می کنند؛ یک پیام تنها زمانی می‌تواند آشکار شود که کلید رمزگشایی، با کلید encryption یکسان باشد.

به‌طورکلی می‌توان گفت الگوریتم‌های رمزنگاری به دو دستهٔ "بر پایهٔ کلید"[5] و غیرکلیدی تقسیم می‌شوند الگوریتم‌های غیرکلیدی هم شامل توابع درهم‌سازی و روشهای رمزنگاری کلاسیک و سنتی هستندکه در مورد توابع درهم سازی صحبت خواهد شد اما روشهای کلاسیک رمزنگاری که قبل از وجود کامپیوترها و ۰ و۱ ها عموماً برای رمزگذاری متون و پیامها مورد استفاده قرار می گرفتند، با امکانات امروزی به راحتی قابل شکستن هستند و در نتیجه نیازی به بحث در مورد آنها نیست.

دو دسته از الگوریتم‌های پنهان‌سازی برپایهٔ کلید موجودند: متقارن[6] و نامتقارن (یاکلید عمومی[7]). تفاوت در اینجاست که الگوریتم‌های متقارن برای encryption و decryption از یک کلید استفاده می کنند (یا کلید آشکارسازی به راحتی از کلید پنهان‌سازی استخراج می‌شود)، در حالیکه الگوریتم‌های نامتقارن برای پنهان سازی وآشکار سازی از کلیدهای متفاوت استفاده می کنند وکلید آشکارسازی نمی‌تواند ازکلید پنهان‌سازی استخراج شود.

الگوریتم‌های متقارن می‌توانند به دو دسته رمزکننده‌های جریانی[8] و رمزکننده‌های بلوکی[9] تقسیم شوند. رمز کننده‌های جریانی می‌توانند در هر زمان یک بیت از plaintext را رمزکنند، در حالیکه رمزکننده‌های بلوکی تعدادی بیت می گیرند (نوعاً ۶۴ بیت در رمزکننده‌های پیشرفته) وآنها را به عنوان یک واحد جدا رمز می کنند.

رمزکننده‌های نامتقارن (الگوریتم‌های کلید عمومی یا بطور کلی رمزنگاری کلید عمومی) اجازه می‌دهند که کلید پنهان سازی، عمومی(آشکار) باشد (حتی می‌تواند در یک روزنامه منتشر شود) تا به هر کس اجازه دهد که با کلید پیامی را رمز

---

[1] Breaking
[2] Cryptographers
[3] Authentication
[4] Cipher
[5] Key-based
[6] Symmetric
[7] Public-key
[8] Stream ciphers
[9] Block ciphers

کند، در حالیکه فقط گیرندهٔ مناسب (کسی که کلید رمزگشایی را می‌داند) می‌تواند پیام را رمزگشایی کند. کلید پنهان‌سازی همچنین کلید عمومی یا همگانی و کلید رمز گشایی، کلید خصوصی[1] یا کلید محرمانه نامیده می‌شود.

الگوریتم‌های رمز نگاری پیشرفته دیگر "کاغذ و قلمی"[2] نیستند. الگوریتم‌های رمزنگاری قوی طراحی شده‌اند تا بوسیلهٔ کامپیوترها یا وسایل سخت افزاری ویژه اجرا شوند. در اکثرکاربردها، رمزنگاری در یک نرم افزار کامپیوتری انجام می‌شود.

عموماً الگوریتم‌های متقارن برای اجرا شدن روی یک کامپیوترخیلی سریعتر از انواع نامتقارن هستند. در عمل، این الگوریتم‌ها غالباً با همدیگر استفاده می‌شوند؛ بنابراین یک الگوریتم کلید عمومی مورد استفاده قرار می‌گیرد تا یک کلید رمزنگاری به صورت تصادفی تولید شده را رمزکند و کلید تصادفی مورد استفاده قرار می‌گیرد تا پیام حقیقی (واقعی) را با استفاده از یک الگوریتم متقارن رمزکند. این عمل گاهی رمز کردن پیوندی (ترکیبی) نامیده می‌شود.

رمزنگاری علاوه بر محرمانگی[3]، همچنین می‌تواند خواص امنیتی زیر را فراهم کند:

- تصدیق[4] : به طرفی که اطلاعات را می‌فرستد اعتبار و رسمیت می‌دهد.
- جامعیت[5]: اطمینان می‌دهد که اطلاعات در هنگام انتقال تغییر نیافته است.
- "عدم انکار"[6]: مانع از انکار یک طرف که پیامی فرستاده یا عملی را انجام داده است، می‌شود.

[1] Private-key
[2] Pencil-and-paper
[3] Confidentiality
[4] Authentication
[5] Integrity
[6] Non-Repudiation

۴-۲. تکنیک‌های رمزنگاری

۴-۲-۱. اهمیت طول کلید در الگوریتم‌های رمزنگاری

امنیت سیستم‌های رمزنگاری وابسته به دو عامل اساسی قدرت الگوریتم و طول کلید می‌باشد. اگر فرض شود که در قدرت الگوریتم هیچ خللی وارد نمی‌شود، هیچ راهی برای شکستن آن غیر از Brute-Force وجود ندارد؛ در این نوع رمزشکنی، تعداد محدودی Plaintext و Ciphertext متناظر با آن وجود دارد و رمزشکن سعی می‌کند تا با آزمایش کلیدهای متفاوت، کلید مطلوب را بیابد. بنابراین، دامنهٔ کلیدهای مورد آزمایش با افزایش طول کلید زیاد می‌شود و شکستن رمز مشکل‌تر می‌گردد.

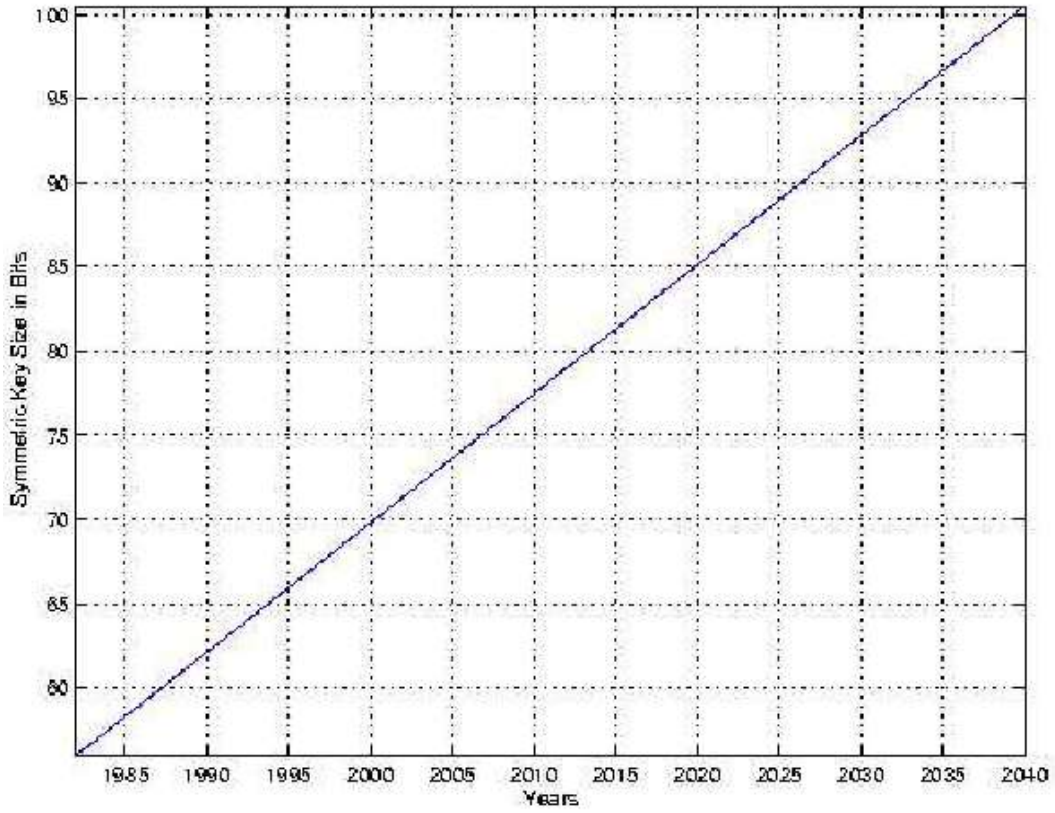
Arjen K. Lenstra و Eric R. Verheul با ارائهٔ فرمولهایی در مقاله‌ای با عنوان " Selecting Cryptographic Key Sizes" در نوامبر ۱۹۹۹، پیشنهادهایی برای حداقل طول کلید در سیستم‌های رمزنگاری متقارن، نامتقارن و خم منحنی ارائه نموده‌اند. در این مقاله، طول کلید برای سیستم‌های رمزنگاری متقارن برابر با اندازهٔ کلید بکارگرفته شده در الگوریتم و برای الگوریتم RSA، "طول پیمانه"[1] و برای لگاریتم گسسته و خمهای منحنی، اندازهٔ میدان (زیرگروه) بکاررفته در الگوریتم می‌باشد. نتایج حاصل از این تحقیق را در جدول ۴-۱ و شکلهای ۴-۱، ۴-۲ و ۴-۳ ملاحظه می‌کنید.

جدول ۴-۱ کران پایین برای اندازهٔ کلیدهای هم‌ارز از لحاظ محاسباتی

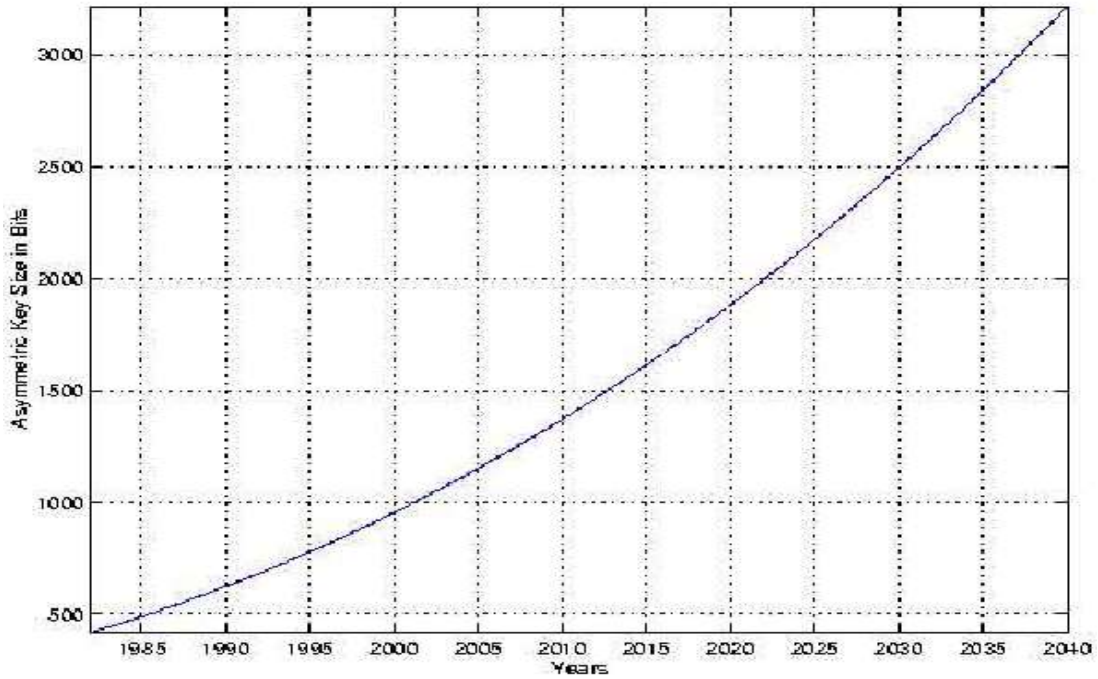
Year	Symmetric Key Size	Classical Asymmetric Key Size (and SDL Field Size)	Subgroup Discrete Logarithm Key Size	Elliptic Curve Key Size		Infeasible number of Mips Years	Lower bound for Hardware cost in US $ for a 1 day attack (cf. (4.5))	A corresponding number of years on 450MHz PentiumII PC
				Progress				
				no	yes			
1999	70	915 672	123	130	130	$4.19 * 10^9$	$1.29 * 10^8$	$9.31 * 10^6$
2000	70	952 704	125	132	132	$7.13 * 10^9$	$1.39 * 10^8$	$1.58 * 10^7$
2001	71	990 736	126	133	135	$1.21 * 10^{10}$	$1.49 * 10^8$	$2.70 * 10^7$
2002	72	1028 768	127	135	139	$2.06 * 10^{10}$	$1.59 * 10^8$	$4.59 * 10^7$
2003	73	1068 800	129	136	140	$3.51 * 10^{10}$	$1.71 * 10^8$	$7.80 * 10^7$
2004	73	1108 832	130	138	143	$5.98 * 10^{10}$	$1.83 * 10^8$	$1.33 * 10^8$
2005	74	1149 864	131	139	147	$1.02 * 10^{11}$	$1.96 * 10^8$	$2.26 * 10^8$
2006	75	1191 896	133	141	148	$1.73 * 10^{11}$	$2.10 * 10^8$	$3.84 * 10^8$
2007	76	1235 928	134	142	152	$2.94 * 10^{11}$	$2.25 * 10^8$	$6.54 * 10^8$
2008	76	1279 960	135	144	155	$5.01 * 10^{11}$	$2.41 * 10^8$	$1.11 * 10^9$
2009	77	1323 1024	137	145	157	$8.52 * 10^{11}$	$2.59 * 10^8$	$1.89 * 10^9$
2010	78	1369 1056	138	146	160	$1.45 * 10^{12}$	$2.77 * 10^8$	$3.22 * 10^9$
2011	79	1416 1088	139	148	163	$2.47 * 10^{12}$	$2.97 * 10^8$	$5.48 * 10^9$
2012	80	1464 1120	141	149	165	$4.19 * 10^{12}$	$3.19 * 10^8$	$9.32 * 10^9$
2013	80	1513 1184	142	151	168	$7.14 * 10^{12}$	$3.41 * 10^8$	$1.59 * 10^{10}$
2014	81	1562 1216	143	152	172	$1.21 * 10^{13}$	$3.66 * 10^8$	$2.70 * 10^{10}$
2015	82	1613 1248	145	154	173	$2.07 * 10^{13}$	$3.92 * 10^8$	$4.59 * 10^{10}$
2016	83	1664 1312	146	155	177	$3.51 * 10^{13}$	$4.20 * 10^8$	$7.81 * 10^{10}$
2017	83	1717 1344	147	157	180	$5.98 * 10^{13}$	$4.51 * 10^8$	$1.33 * 10^{11}$
2018	84	1771 1376	149	158	181	$1.02 * 10^{14}$	$4.83 * 10^8$	$2.26 * 10^{11}$
2019	85	1825 1440	150	160	185	$1.73 * 10^{14}$	$5.18 * 10^8$	$3.85 * 10^{11}$
2020	86	1881 1472	151	161	188	$2.94 * 10^{14}$	$5.55 * 10^8$	$6.54 * 10^{11}$
2021	86	1937 1536	153	163	190	$5.01 * 10^{14}$	$5.94 * 10^8$	$1.11 * 10^{12}$
2022	87	1995 1568	154	164	193	$8.52 * 10^{14}$	$6.37 * 10^8$	$1.89 * 10^{12}$
2023	88	2054 1632	156	166	197	$1.45 * 10^{15}$	$6.83 * 10^8$	$3.22 * 10^{12}$

[1] Modulus size

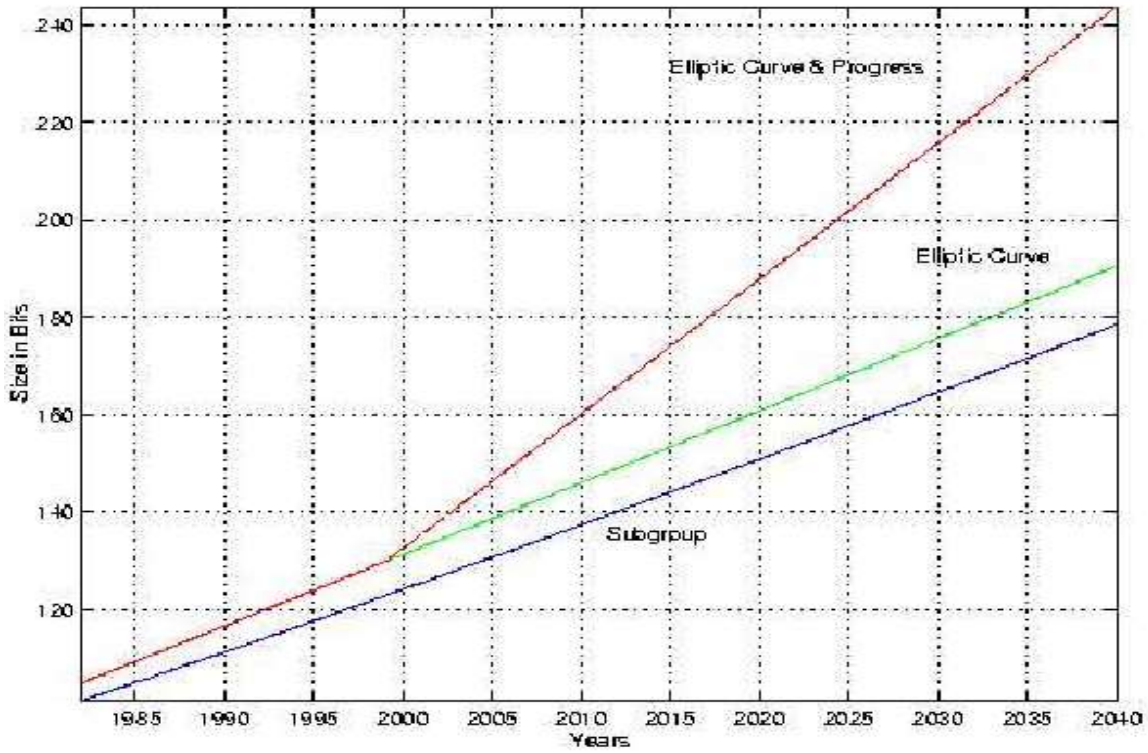
2024	89	2113_{1696}	157	167	198	$2.47 * 10^{15}$	$7.32 * 10^{8}$	$5.48 * 10^{12}$
2025	89	2174_{1728}	158	169	202	$4.20 * 10^{15}$	$7.84 * 10^{8}$	$9.33 * 10^{12}$
2026	90	2236_{1792}	160	170	205	$7.14 * 10^{15}$	$8.41 * 10^{8}$	$1.59 * 10^{13}$
2027	91	2299_{1856}	161	172	207	$1.21 * 10^{16}$	$9.01 * 10^{8}$	$2.70 * 10^{13}$
2028	92	2362_{1888}	162	173	210	$2.07 * 10^{16}$	$9.66 * 10^{8}$	$4.59 * 10^{13}$
2029	93	2427_{1952}	164	175	213	$3.52 * 10^{16}$	$1.04 * 10^{9}$	$7.81 * 10^{13}$
2030	93	2493_{2016}	165	176	215	$5.98 * 10^{16}$	$1.11 * 10^{9}$	$1.33 * 10^{14}$
2031	94	2560_{2080}	167	178	218	$1.02 * 10^{17}$	$1.19 * 10^{9}$	$2.26 * 10^{14}$
2032	95	2629_{2144}	168	179	222	$1.73 * 10^{17}$	$1.27 * 10^{9}$	$3.85 * 10^{14}$
2033	96	2698_{2208}	169	181	223	$2.95 * 10^{17}$	$1.37 * 10^{9}$	$6.55 * 10^{14}$
2034	96	2768_{2272}	171	182	227	$5.01 * 10^{17}$	$1.46 * 10^{9}$	$1.11 * 10^{15}$
2035	97	2840_{2336}	172	184	230	$8.53 * 10^{17}$	$1.57 * 10^{9}$	$1.90 * 10^{15}$
2036	98	2912_{2400}	173	185	232	$1.45 * 10^{18}$	$1.68 * 10^{9}$	$3.22 * 10^{15}$
2037	99	2986_{2464}	175	186	235	$2.47 * 10^{18}$	$1.80 * 10^{9}$	$5.49 * 10^{15}$
2038	99	3061_{2528}	176	188	239	$4.20 * 10^{18}$	$1.93 * 10^{9}$	$9.33 * 10^{15}$
2039	100	3137_{2592}	178	189	240	$7.14 * 10^{18}$	$2.07 * 10^{9}$	$1.59 * 10^{16}$
2040	101	3214_{2656}	179	191	244	$1.22 * 10^{19}$	$2.22 * 10^{9}$	$2.70 * 10^{16}$



شکل ۴-۱ حداقل طول کلید پیشنهادی برای سیستم‌های رمزنگاری متقارن



شکل ۴-۲ حداقل طول کلید پیشنهادی برای سیستم‌های رمزنگاری نامتقارن کلاسیک



شکل ۴-۳ حداقل طول کلید پیشنهادی برای زیرگروه لگاریتم گسسته و سیستم‌های خم منحنی

۴-۲-۲. مدیریت کلید

بسیاری از حملات علیه الگوریتم‌های متقارن و نامتقارن بر روی مدیریت کلید انجام می‌گیرد. مدیریت کلید شامل عملیات تولید، انتقال و نگهداری کلید می‌باشد و نگهداری کلید شامل عملیات بروزرسانی، ذخیره و پشتیبان‌گیری از کلید می‌باشد.

۴-۲-۲-۱. تولید کلید

برای تولید کلید به صورت تصادفی، بهترین روش، استفاده از "مولدهای اعداد شبه‌تصادفی" می‌باشد. این مولدها توابع یکطرفه‌ای می‌باشند که از یک عدد تصادفی کوچک، رشتهٔ تصادفی بزرگتری می‌سازند؛ به‌نحویکه حدس‌زدن عدد تصادفی تولید شده بسیار مشکل می‌باشد. استاندارد ANSI X9.17(تجدیدنظرشده) یک روش برای تولید کلیدهای تصادفی درون یک سیستم پیشنهاد نموده‌است. در این روش، از الگوریتم رمزنگاری DES استفاده می‌شود.

در RSA، برای تولید کلیدهای عمومی و خصوصی ابتدا دو عدد اول بزرگ انتخاب می‌شود و ازاین دو عدد کلیدها ساخته می‌شوند؛ در سایر الگوریتم‌های نامتقارن نیز تولید کلید بر اساس تولید عدد اول P برای اندازهٔ میدان می‌باشد.

۴-۲-۲-۲. انتقال کلید

در الگوریتم‌های متقارن، کلید تولید شده باید به صورت امن به طرف مقابل انتقال یابد. از روشهای معمول انتقال کلیدهای الگوریتم‌های متقارن استفاده از الگوریتم‌های نامتقارن و روش رمزنگاری کوانتومی می‌باشد. البته رمزنگاری کوانتمی در مراحل تحقیقاتی و آزمایشگاهی قرار دارد و محصول تجاری از آن تا کنون وارد بازار نشده است. در آزمایشی در گروه فیزیک دانشگاه جنوا در سال ۲۰۰۰ توانسته‌اند کلیدهای ۲۰ کیلو بیتی را به مسافت ۲۳ کیلومتر با استفاده از فیبرنوری و با نرخ خطای ۱/۳۵ درصدی ارسال کنند. این رمزنگاری بر اساس قوانین کوانتوم استوار است و تضمین می‌کند که کلید منتقل‌شده با استفاده از این روش، قابل کشف توسط شخص سومی نیست.

الگوریتم‌های نامتقارن نیز یکی از روشهای انتقال کلید می‌باشند، با استفاده از کلید عمومی طرف مقابل، داده‌ها رمز و فرستاده می‌شوند.

از الگوریتم‌های متقارن نیز می‌توان برای رمز کلیدها استفاده نمود. بدین ترتیب، دو نوع کلید درسیستم وجود خواهد داشت؛ کلیدهایی برای رمزکردن کلیدهای الگوریتم متقارن و کلیدهایی نیز برای رمزکردن داده. برای استفاده از این راه‌حل، باید کلیدهای رمزکلید به‌صورت دستی پخش شوند.

یک راه‌حل، تکه‌تکه‌کردن کلید و فرستادن جداگانهٔ هریک از قسمتها بر روی کانال‌های متفاوت است؛ برای مثال یک بخش بر روی خط تلفن، یک بخش توسط نامهٔ الکترونیکی و بخشی نیز می‌تواند توسط پست انتقال یابد.

طرف دریافت‌کنندهٔ کلید، باید از صحت کلید منتقل‌شده اطمینان یابد؛ رمزنگاری کوانتومی قراردادهایی را برای بررسی صحت کلید منتقل‌شده دارد. در الگوریتم‌های نامتقارن، شخص رمزکنندهٔ کلید باید به صحت کلید عمومی طرف مقابل اطمینان یابد که این کار از طریق CA امکان‌پذیر است؛ طرف دریافت‌کنندهٔ کلید نیز باید به صحت فرستندهٔ آن مطمئن باشد. دراین حالت، شخص رمزکننده می‌تواند از امضای دیجیتالی استفاده کند. در رمزکردن کلید با استفاده از رمزنگاری متقارن نیز باید مطمئن بود که شخص دیگری به کلید دسترسی ندارد. برای بررسی خطاهای انتقال نیز می‌توان از توابع درهم‌سازی استفاده نمود.

۴-۲-۲-۳. نگهداری کلید

بعد از مرحلهٔ انتقال، کلیدها در سیستم باید به‌درستی نگهداری شوند؛ نگهداری صحیح شامل به‌روزرسانی به‌موقع کلیدها، ذخیرهٔ امن آنها و پشتیبان‌گیری از کلیدها می‌باشد. برای مثال اگرسیستمی بخواهد روزانه کلیدها را به‌روز کند بهترین روش این است که طرفین از کلید قدیمی، کلید جدید را تولید کنند. به‌روزرسانی کلید به معنی تغییر کلید با استفاده از یک فرآیند غیرقابل برگشت می‌باشد. برای این کار، یک تابع یکطرفه لازم است که توسط آن بتوان از کلید قدیمی کلید جدید را بدست آورد. امنیت کلید جدید به همان اندازهٔ امنیت کلید قدیمی خواهد بود. درحقیقت اگر طرف سومی به کلید قدیمی دسترسی داشته باشد، می‌تواند کلید جدید را نیز تولید کند.

ذخیرهٔ کلید نیز باید به‌صورت امن، ممکن باشد. امروزه کارتهای هوشمند و حافظه‌های فقط-خواندنی که بخشی از کلید را حمل می‌کنند، ابزارهای مطمئنی برای ذخیرهٔ کلیدها هستند. روشهای دیگری ازجمله ذخیره‌سازی بر روی دیسک سخت کامپیوتر نیز برای ذخیرهٔ کلید وجود دارند که مزایایی بهتر از کارتهای هوشمند ندارند. کارتهای هوشمند یک وسیلهٔ قابل حمل هستند که کمتر از روشهای دیگر آسیب‌پذیر می‌باشند.

در صورتیکه دیسک سخت برای ذخیرهٔ کلید خصوصی بکار رود، با ریسکهایی همراه است. ازجمله اینکه همواره احتمال Crash کردن دیسک سخت و از بین‌رفتن داده‌ها وجود دارد. کپی‌کردن اطلاعات از دیسک سخت بسیار آسان است در حالیکه کپی داده از کارتهای هوشمند به‌راحتی امکان‌پذیر نیست. پشتیبان‌گرفتن از کلیدها برای استفاده از آنها در هنگامی‌که سیستم نگهدارندهٔ کلید دچار مشکل شود ضروری است. برای پشتیبان‌گرفتن از کلید می‌توان از انواع روشهای پشتیبان‌گیری و بازیابی استفاده کرد. پشتیبان‌گیری مخصوصاً برای زمانیکه کلیدها بر روی دیسک سخت ذخیره می‌شوند، ضروری می‌باشد.

۴-۲-۳. مودهای کاری رمزکننده‌های بلوکی

۴-۲-۳-۱. تعاریف و کارهای صورت‌گرفته یا در حال انجام

یک "مود کاری"[1] یا به اختصار، مود، یک الگوریتم است که استفاده از یک الگوریتم رمز بلوکی کلید متقارن جهت مهیاکردن یک سرویس اطلاعی نظیر محرمانگی یا اعتبارسنجی را برجسته می‌کند. با ظهور رمزکننده‌های بلوکی جدید، نظیر استاندارد رمزنگاری پیشرفته (AES) نیاز است که مودهای عمل قدیمی به‌روز شوند و در این فرصت برای ایجاد مودهای جدید، ملاحظاتی صورت گیرد. برای تسهیل ورود عمومی برای ایجاد مودهای عمل، NIST یک کارگاه عمومی در اواخر ۲۰۰۰، و یک کارگاه عمومی دوم در تابستان ۲۰۰۱ برگزار کرد. همچنین NIST درحال مشخص‌کردن مودها در یک سری از نشریات ویژه می‌باشد. اولین سند در سری اسناد، پنج مود محرمانگی را مشخص می‌کند و سند دوم یک مود اعتبارسنجی را مشخص خواهد کرد.

سپس NIST بررسی خواهد کرد که آیا مودهای اضافی نیز باید مشخص گردند یا خیر. چندین پیشنهاد برای مودها شامل مودهایی برای اعتبارسنجی، رمزکردن تصدیق‌شده، درهم‌سازی، و تولید بیت تصادفی به NIST ارائه شده‌اند.

۴-۲-۳-۲. توصیهٔ NIST برای مودها

در یک سند ویژه (SP 800-38A) با عنوان توصیه‌ای برای مودهای کاری رمزکنندهٔ بلوکی، پنج مود محرمانگی برای استفاده با هر رمزکنندهٔ بلوکی پذیرفته شده به عنوان FIPS نظیر AES، مشخص می‌شوند. مودهای داخل SP 800-38A، نسخه‌های به‌روزآورده‌شدهٔ مودهای ECB، CBC، CFB و OFB که در FIPS Pub. 81 مشخص شده‌اند، می‌باشند؛ به علاوه، SP 800-38A مود CTR را معرفی می‌کند.

NIST انتظار دارد که یک ویرایش ۲۰۰۲ از SP 800-38A را منتشر کند که در آن، دامنهٔ مود CBC گسترش می‌یابد (جهت شامل‌کردن Plaintext هایی که طول بیت آنها مضربی از اندازهٔ بلوک نمی‌باشد).

SP 800-38B یک اصلاح از مود اعتبارسنجی CBC-MAC را برای استفاده با هر رمزکنندهٔ بلوکی پذیرفته‌شده به عنوان FIPS مشخص خواهد کرد.

۴-۲-۳-۳. مودهای کاری رمزکننده‌های بلوکی

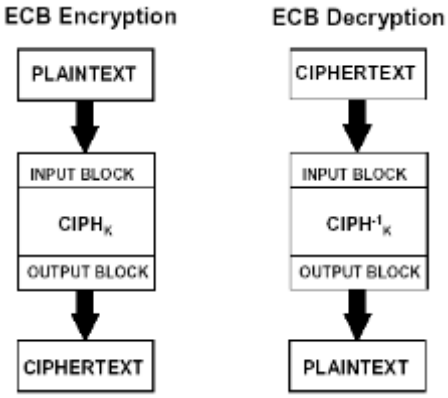
در ادامهٔ مطالب از کلمهٔ CIPH بارها استفاده شده‌است که در زیر به مفهوم آن اشاره می‌شود.

- $CIPH_K(X)$ تابع رمزکنندهٔ الگوریتم رمزکنندهٔ بلوکی تحت کلید K به بلوک دادهٔ X اعمال می‌شود.
- $CIPH\text{-}1_K(X)$ تابع معکوس رمزکنندهٔ الگوریتم رمزکنندهٔ بلوکی تحت کلید K به بلوک دادهٔ X اعمال می‌شود.

[1] Mode of operation

◆ مود ECB[1]

Codebook یک کتاب حاوی یک لیست الفبایی از کلمات یا عبارات به‌همراه کد معادل آنها می‌باشد. این مود، ساده‌ترین و بدیهی‌ترین کاربرد یک رمزکنندهٔ بلوکی می‌باشد بدین‌صورت که کلید محرمانه برای رمزکردن بلوک Plaintext جهت تشکیل یک بلوک Ciphertext استفاده می‌شود. بنابراین، دو بلوک Plaintext یکسان همیشه یک بلوک Ciphertext را تولید خواهد کرد. هرچندکه این مود، معمولی‌ترین مود رمزکننده‌های بلوکی می‌باشد، نسبت به حملات گوناگون Brute-Force آسیب‌پذیر است. شکل ۴-۴، نحوهٔ عمل مود ECB را نشان می‌دهد.



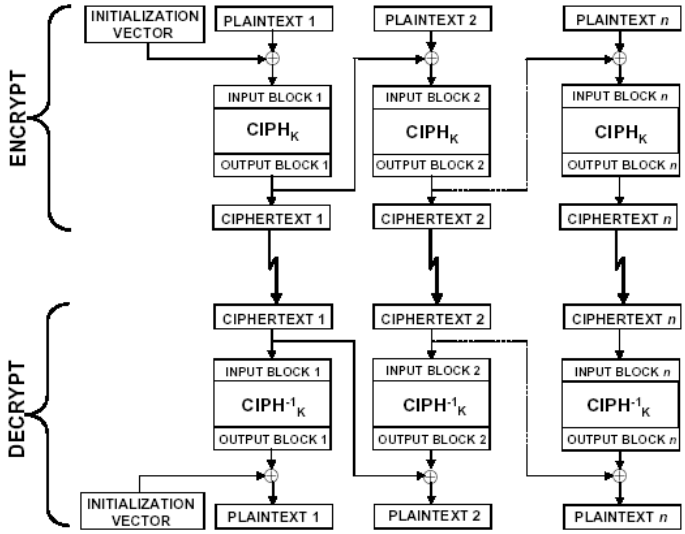
شکل ۴-۴ مود ECB

◆ مود CBC[2]

در این مود، یک مکانیزم پس‌خورد به روش رمزکردن اضافه می‌شود. در CBC، Plaintext با بلوک قبلی Ciphertext پیش از رمزکردن یای انحصاری (XOR) می‌شود. نتیجهٔ این عمل، این است که دو بلوک یکسان Plaintext، هرگز به یک Ciphertext رمز نمی‌شوند.

مود CBC را در شکل ۴-۵ می‌توان مشاهده کرد.

[1] Electronic CodeBook
[2] Cipher Block Chaining

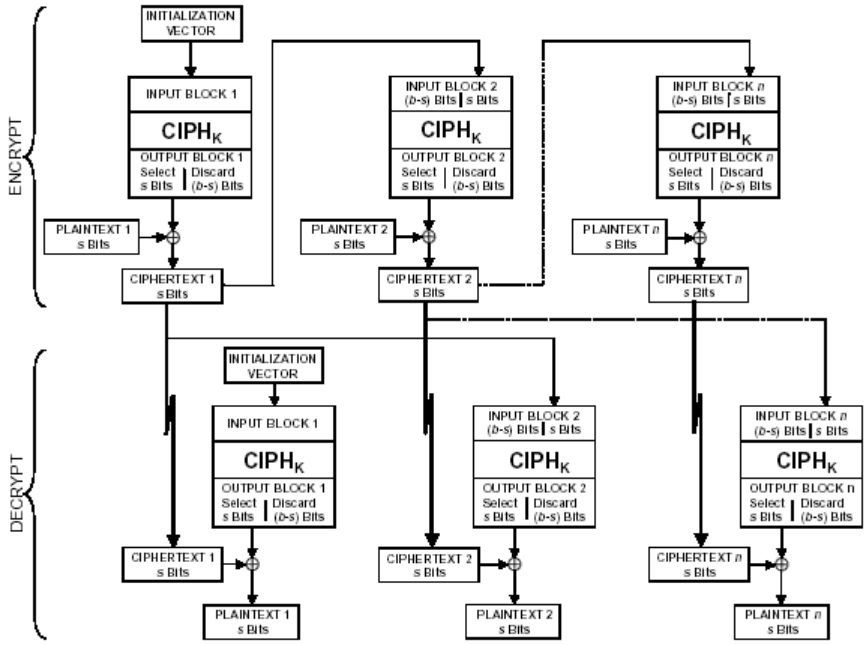


شکل ۴- ۵ مود CBC

◆ مود CFB[1]

یک پیاده‌سازی رمزکنندهٔ بلوکی به عنوان یک رمزکنندهٔ جریانی "خودهمگام"[2] می‌باشد. مود CFB، به داده اجازه می‌دهد که در واحدهایی کوچکتر از اندازهٔ بلوک رمز شود. این عمل ممکن است در بعضی کاربردها نظیر رمزکردن ورودی محاوره‌ای ترمینال مفید باشد. به عنوان مثال، اگر ما از مود CFB یک‌بایتی استفاده می‌کردیم، هر کاراکتر واردشونده، درون یک ثبات به همان اندازهٔ بلوک جای می‌گیرد، رمز می‌شود وبلوک منتقل می‌گردد. در طرف گیرنده، Ciphertext رمزگشایی می‌شود و بیتهای اضافی موجود در بلوک (یعنی، هر چیزی بیشتر از یک بایت کفایت‌کننده)، دور انداخته می‌شوند. شکل ۴-۶ نحوهٔ عمل، در این مود کاری را نشان می‌دهد.

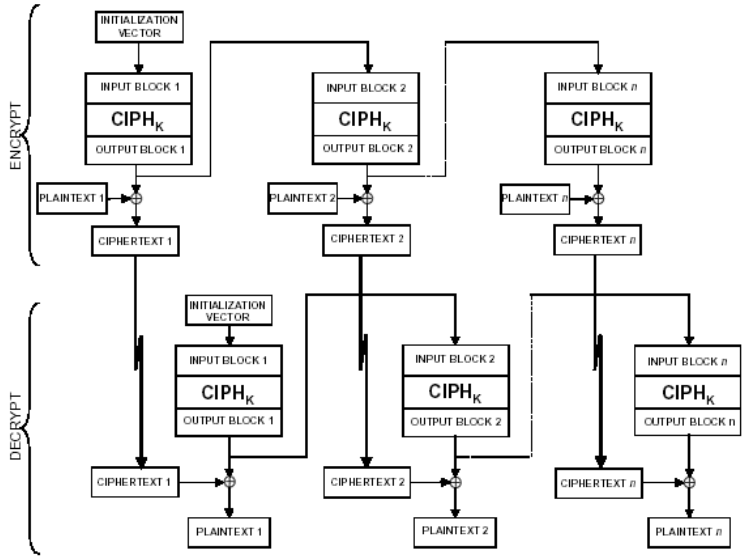
[1] Cipher FeedBack
[2] Self-synchronizing



شکل ۴-۶ مود CFB

◆ مود OFB[1]

این مود، یک پیاده‌سازی رمزکنندهٔ بلوکی به صورت مجازی شبیه یک رمزکنندهٔ جریانی همگام می‌باشد. OFB از تولید یک بلوک Ciphertext از یک بلوک Plaintext بوسیلهٔ یک مکانیزم پس‌خورد داخلی که مستقل از جریان بیت Plaintext و Ciphertext است؛ جلوگیری می‌کند (در شکل ۴-۷، نحوهٔ عمل OFB نشان داده شده‌است).

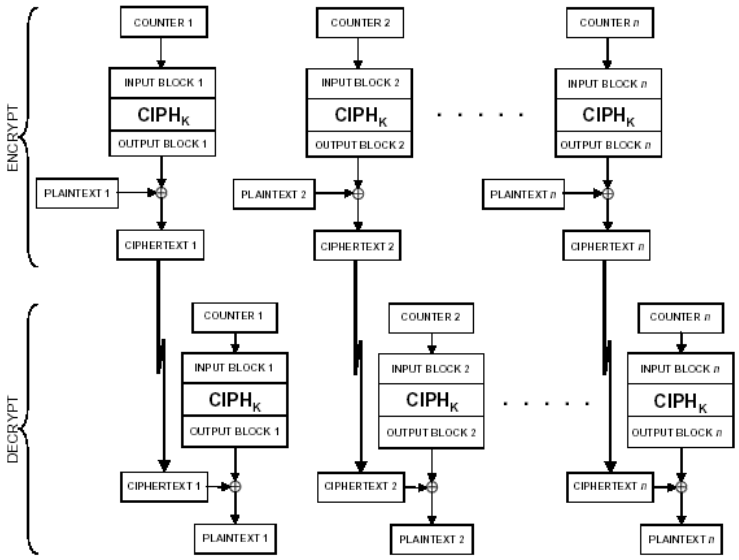


شکل ۴-۷ مود OFB

[1] Output FeedBack

- **مود CTR**[1]

در این مود، رمزکننده به یک مجموعه از بلوکهای ورودی به نام شمارنده‌ها جهت تولید توالی از بلوکهای خروجی اعمال می‌شود که برای تولید Ciphertext با Plaintext یای انحصاری می‌شوند (شکل ۴-۸).



شکل ۴-۸ مود CTR

۴-۲-۴. کاربردهای الگوریتم‌ها

دراین بخش، به کاربرد الگوریتم‌های رمزنگاری در پروتکلهای مختلف می‌پردازیم که این پروتکل‌ها هر یک در کاربردهای خاص خود به‌کار می‌روند. رمزنگاری درسطوح متفاوتی به‌کار می‌رود. در پائین‌ترین سطح، الگوریتم‌هایی از قبیل الگوریتم‌های بلوکی و سیستم‌های رمزنگاری کلید عمومی وجود دارند. بر روی این سطح، پروتکل‌هایی قرار دارند که از الگوریتم‌های رمزنگاری استفاده می‌کنند. در سطح بالاتراز پروتکل‌ها برنامه‌های کاربردی قراردارند که از پروتکلها استفاده می‌کنند. به همین علت مطالعهٔ امنیت الگوریتم‌ها به تنهایی کافی نیست بلکه ضعف یک پروتکل یا کاربرد لایهٔ بالاتر می‌تواند کل سیستم را ناامن کند؛ بدون توجه به اینکه الگوریتم‌های لایهٔ پائین‌تر چقدر امن هستند. برای مثال اگر پروتکلی نتواند به صورت امن کلیدهای رمزنگاری را انتقال دهد، هرچه الگوریتم خوب عمل کند، سیستم ناامن خواهد شد. آنالیز پروتکل‌ها نیز اغلب کارمشکلی است چراکه پیاده‌سازیهایی که از پروتکلها انجام می‌شوند ممکن است منجر به مشکلات بیشتری شوند. یک پروتکل خوب به تنهایی کافی نیست بلکه پیاده‌سازیهای خوب و محکمی نیز لازم دارد. چند پروتکل مطرح و استانداردشده که از الگوریتم‌های رمزنگاری استفاده می‌کنند در زیر آورده شده‌اند.

[1] Counter

۴-۲-۴-۱. DNSSec[1]

DNSSec، پروتکلی برای امنیت DNS ها می‌باشد. DNS، یک پایگاه دادهٔ توزیع‌شدهٔ سلسله مراتبی است که سرویسهای مفیدی را ارائه می‌کند. از جملهٔ آنها ترجمهٔ آدرسهای IP به نامهای host قابل خواندن برای انسانها می‌باشد. بخاطر اهمیت اطلاعات ذخیره‌شده توسط DNS نیاز شدیدی برای امنیت ارتباطات درون سیستم DNS وجود دارد. DNS به طورمعمول از حملاتی که پیغامهایی را به سیستم DNS تزریق می‌کنند و یا آن را تغییر می‌دهند جلوگیری نمی‌کند. یک حملهٔ معمول به DNS، DNS Spoofing می‌باشد که در آن، حمله‌کننده جوابهای DNS را در سر راهش به کاربر دستکاری می‌کند. اگر یک حمله‌کننده بتواند جدول DNS را در یک سرویس‌دهنده تغییر دهد، آن تغییرات درکل اینترنت پخش خواهد شد. برای مثال، اگر کاربری بخواهد به سایت CNN دسترسی پیدا کند، یک سرویس‌دهندهٔ DNS دستکاری‌شده می‌تواند کاربر را به یک ماشین بدخواه بفرستد که محتوای آن با سایت CNN متفاوت است.

برای حفظ امنیت در سرورهای DNS، IETF راه‌حلی را با نام DNSSec پیشنهاد نموده است. DNSSec روشی برای استفاده از تکنیکهای رمزنگاری در پیغامهای DNS می‌باشد. فرد درخواست‌کنندهٔ اطلاعات از امضاهای دیجیتالی برای صحت مبدأ اطلاعات استفاده می‌کند. بدین‌ترتیب، پیغام‌ها قابل ردیابی است. DNSSec، دو رکورد برای تصدیق به DNS اضافه می‌کند؛ رکوردهای KEY و SIG. رکورد KEY کلید عمومی را برای یک host یا یک ناحیهٔ اجرایی اضافه می‌کند و رکورد SIG یک امضای دیجیتالی را در ارتباط با هر مجموعه از رکوردها ذخیره می‌کند. یک DNSSEC می‌تواند رکوردهای امضاءشده را از رکوردهای امضاء نشده تشخیص دهد و در مواقع لزوم برای رکوردهای امضاء شده یک خطا را گزارش کند.

۴-۲-۴-۲. GSSAPI[2]

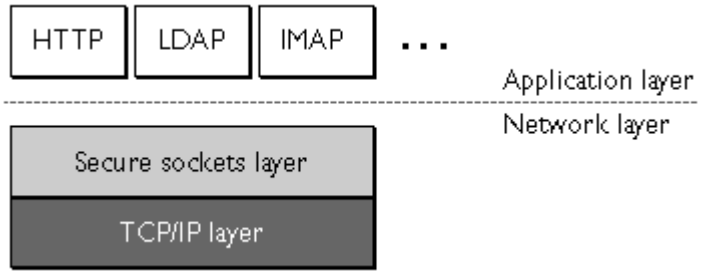
GSSAPI یک مکانیزم تعیین هویت، مبادلهٔ کلید می‌باشد و یک واسط رمزنگاری از الگوریتم‌ها و سیستم‌های رمزنگاری متفاوت ارائه می‌دهد. GSSAPI یک API[3] عمومی است که برای تعیین هویت‌های سرویس‌دهنده-سرویس‌گیرنده بکار می‌رود. انگیزهٔ پدیدآمدن چنین APIای این است که هر سیستم امنیتی، API مخصوص به خودش را دارد و اضافه‌کردن سیستم‌های امنیتی به کاربردها با وجود APIهای متفاوت امنیتی که دارند؛ مشکل می‌باشد. بنابراین، لزوم یک API مشترک برای سرویسهای امنیتی لازم به نظر می‌رسد.

۴-۲-۴-۳. SSL

SSL یکی از دو پروتکل امن برای امنیت وب می‌باشد (دیگری SHTTP می‌باشد). امنیت وب به‌علت کاربرد همه‌گیرآن بسیار مهم می‌باشد. SSL برای اولین بار توسط Netscape به‌عنوان یک پروتکل استاندارد توسعه یافت. اولین هدف SSL تأمین یک ارتباط مخفی و قابل اعتماد بین کاربردها درسطح اینترنت می‌باشد.

[1] Domain Name Server Security
[2] Generic Security Service API
[3] Application Programming Interface

SSL یک پروتکل برای انتقال پیام در اینترنت می‌باشد که بین پروتکلهای لایهٔ شبکه و پروتکلهای لایه کاربرد قرار می‌گیرد (شکل ۴-۹) و تضمین می‌کند که ارتباطات بر روی اینترنت به صورت امن برقرار می‌شود.



شکل ۴-۹ SSL بر روی لایهٔ TCP/IP و زیر پروتکل‌های کاربردای لایهٔ بالاتر اجرا می‌شود.

SSL وظایف زیر را انجام می‌دهد:

- اعتبارسنجی سرویس‌دهنده : اجازه می‌دهد تا یک کاربر، هویت یک سرویس‌دهنده را تأیید کند. نرم‌افزار Client می‌تواند از تکنیک‌های استاندارد کلید عمومی استفاده کند، تا گواهی یک سرور و id عمومی آن را که توسط یک CA منتشر شده‌است؛ بررسی کند.
- اعتبارسنجی سرویس‌گیرنده : اجازه می‌دهد که یک سرویس‌دهنده هویت یک کاربر را تصدیق کند.

همهٔ اطلاعات فرستاده‌شده بین یک Client و یک سرور توسط نرم‌افزار فرستندهٔ اطلاعات و نرم‌افزار گیرندهٔ اطلاعات رمز می‌شود. پروتکل SSL دارای دو بخش اصلی است: SSL handshake protocol و SSL record protocol. SSL Record Protocol، فرمت داده‌های ارسالی را مشخص می‌کند. SSL handshake protocol، پروتکلی است که یک سری پیغامهای ارتباطی بین سرور با قابلیت SSL وClient با قابلیت SSL برای برقرارکردن یک ارتباط اولیه SSL رد و بدل می‌کند.

این پیغامها شامل موارد زیر است :

- شناسایی سرور توسط Client
- اجازه به سرور و Client برای انتخاب الگوریتم‌های رمزنگاری
- شناسایی دلخواهی Client توسط سرور
- استفاده از تکنیک‌های رمزنگاری کلید عمومی برای تولید رمزهای مشترک
- بوجود آوردن یک ارتباط SSL رمزشده

SSL برای اهداف متفاوتی از الگوریتم‌های رمزنگاری استفاده می‌کند از جملهٔ این اهداف، تصدیق هویت سرور وclient، انتقال گواهی‌ها و نشاندن کلیدهای جلسه می‌باشد.

درhandshaking، پروتکل SSL سرور و client برای انتخاب الگوریتم رمز با هم توافق می‌کنند. این الگوریتم‌های رمز شامل است بر:

- DES الگوریتم استاندارد رمز بلوکی
- DSA الگوریتم امضاهای دیجیتالی
- KEA الگوریتم برای معاوضهٔ کلید
- MD5
- RC2 & RC4
- RSA یک الگوریتم رمز کلیدعمومی که هم برای رمزنگاری و هم برای تصدیق هویت بکار می‌رود.
- RSA Key Exchange که یک الگوریتم مبادلهٔ کلید برپایهٔ الگوریتم RSA می‌باشد.
- SHA-1
- SKIPJACK یک الگوریتم متقارن است که در سخت‌افزار FORTEZIA نیز پیاده‌سازی شده‌است.
- Triple-DES

هنگامی‌که Client و سرور تحت پروتکل SSL به handshake می‌پردازند، درطول handshake، بهترین رمز را بسته به حساسیت داده‌هایی که باید منتقل شوند، سرعت رمز و قابلیت اجراء قوانین، انتخاب می‌کنند. بعضی از سازمان‌ها ممکن است مانع از ارتباط SSL با رمزهای ضعیف‌تر شوند.

جدول ۴-۲ رمزهایی را نشان می‌دهد که هم در SSL2.0 و هم در SSL3.0 پشتیبانی می‌شوند.

جدول ۴- ۲ الگوریتم‌های مورد استفاده در SSL

قدرت الگوریتم‌ها	الگوریتم‌های رمزنگاری
قویترین رمز: فقط برای ایالات متحده اجازه پیاده‌سازی داده شده‌است. این رمز برای بانک‌ها و انستیتوهایی که حامل داده‌های مهمی هستند بکار می‌رود.	**3DES** که ۱۶۸ بیتی است و همراه با الگوریتم SHA1 برای چک صحت پیام بکار می‌رود. قویترین رمزی است که توسطSSL پشتیبانی می‌شود. اما به سرعت RC4 نیست. هم SSL2.0 و هم SSL3.0 از این الگوریتم استفاده می‌کنند.

رمزهای‌قوی: فقط در ایالات متحده اجازه پیاده‌سازی دارند. این رمز برای نیازهای دولتی وکاری به اندازهٔ‌کافی قوی می‌باشد.	**RC4** با ۱۲۸ بیت کلید و همراه با الگوریتم MD5 برای چک صحت پیام بکار می‌رود. چون الگوریتم‌های RC4 وRC2، ۱۲۸ بیتی هستند، قویترین الگوریتم‌ها پس از 3DES می‌باشند. رمزهای RC4 سریع‌ترین رمزهای پشتیبانی‌شده توسط SSL هستند. هم SSL3.0 و هم SSL2.0 این رمزها را پشتیبانی می‌کنند.
	RC2 با ۱۲۸ بیت کلید همراه با الگوریتم MD5 همانند RC4 استفاده می‌شود، اما کندتر از آن است.
	DES با ۵۶ بیت کلید همراه با الگوریتم SHA-1 استفاده می‌شود. DES الگوریتم قویی می‌باشد اما از سه الگوریتم بالا بعلت کمتربودن تعداد کلیدهای ممکن ضعیف‌تر است. هر دو نسخهٔ SSL از DES استفاده می‌کنند.
رمزهای قابل صدور: این رمزها به قدرت رمزهای بالا نیستند اما می‌توان در بیشتر کشورها از آنها استفاده کرد (امروزه به‌راحتی قابل شکستن هستند).	RC4 با ۴۰ بیت کلید و الگوریتم تصدیق صحت RC4. MD5 سریع‌ترین رمز پشتیبانی‌شده می‌باشد.
	RC2 با ۴۰ بیت کلید و الگوریتم تصدیق صحت MD5. RC2 از RC4 کندتر می‌باشد.
ضعیف‌ترین رمز: که در آن از هیچ رمزی استفاده نمی‌شود و فقط از الگوریتم درهم‌سازی MD5 استفاده می‌شود. درهنگام استفاده از آن باید توجه داشت که بعلت رمزنشدن داده، همواره احتمال افشای اطلاعات وجود دارد.	بدون رمز با الگوریتم تصدیق صحت MD5. در این روش تنها می‌توان دستکاری داده را در طول انتقال تشخیص داد. تنها در SSL3.0 پشتیبانی شده‌است.

۴-۴-۲-۴. SHTTP[1]

SHTTP پروتکلی دیگر برای تأمین امنیت در تراکنش‌های وب می‌باشد و در بسیاری از جاها قابل انعطاف‌تر از SSL عمل می‌کند. SHTTP پروتکلی است که برای محافظت از تراکنش‌های HTTP طراحی شده‌است. این قرارداد از مکانیزم‌های متفاوتی برای محرمانگی، تصدیق هویت و درستی استفاده می‌کند. این پروتکل، یک ارتباط امن را بین جفت‌های Client-Server که از پروتکل HTTP استفاده می‌کنند؛ برقرار می‌کند و سعی می‌کند مکانیزم‌های مدیریت کلید، مدل‌های امن، الگوریتم‌های رمزنگاری و فرمت‌های encapsulation را بین طرفهای تراکنش برقرارکند. S-HTTP اجازه می‌دهد که پیغام‌ها به طرق مختلف encapsulate شوند. encapsulation می‌تواند شامل رمزنگاری، امضاء و یا

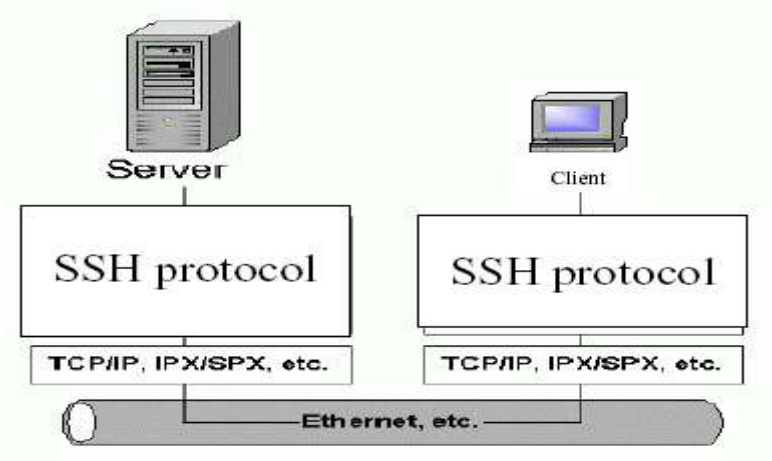
[1] Secure- HTTP

تصدیق برپایهٔ MAC باشد. همچنین S-HTTP شامل سرآیندهای مختلفی برای انتقال کلید، انتقال گواهی و توابع مدیریتی مشابه می‌باشد.

۴-۲-۴-۵. پروتکل‌های SSH1[1] و SSH2

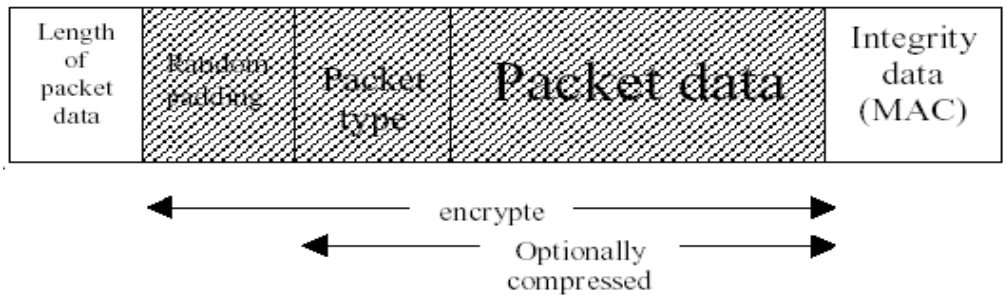
این پروتکل‌ها توسط IETF توسعه یافته‌اند. SSH2 بسیاری از نیازهای اینترنت را برطرف می‌کند و هم اکنون در نرم‌افزار SSH2 استفاده می‌شود. این پروتکل، برای امن‌نمودن نشست‌ها[2] و ارتباطات اختیاری TCP استفاده می‌شود. پروتکل SSH2 توسعه‌یافته‌ای از روایت قبلی‌اش (SSH1) می‌باشد.

SSH1 و SSH2 پروتکل‌های متفاوتی هستند که در ذیل به خصوصیات هریک می‌پردازیم. SSH1 پروتکلی است که بر روی هر لایهٔ انتقالی عمل می‌کند. شکل ۴-۱۰ محل قرارگرفتن قرارداد SSH1 را نشان می‌دهد.



شکل ۴-۱۰ پروتکل SSH1

رمزنگاری در این قرارداد، توسط رمزهای بلوکی نظیرIDEA (مود CBC) یا 3DES (مود CBC) انجام می‌شود. شکل ۴-۱۱ محتویات یک بستهٔ SHH1 را نشان می‌دهد.



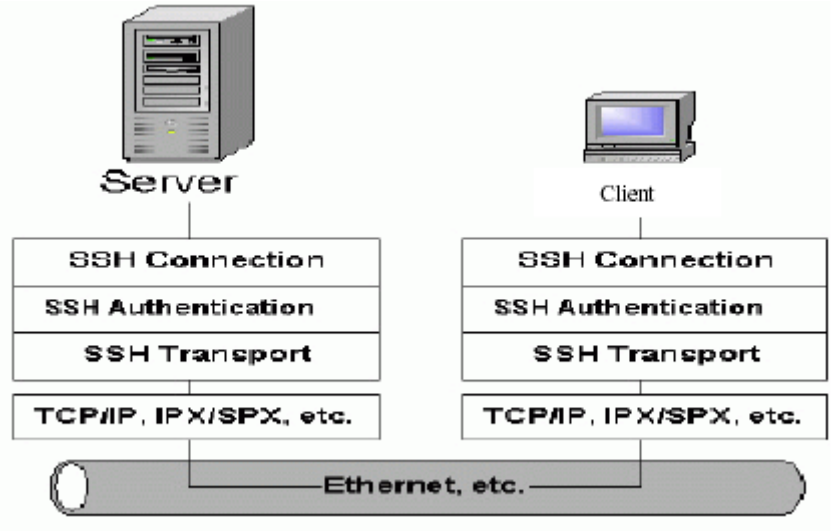
شکل ۴-۱۱ محتویات یک بستهٔ SSH1

[1] Secure Shell
[2] Session

مکانیزم مبادلهٔ کلید در SSH1 برپایهٔ الگوریتم RSA انجام می‌شود. تصدیق هویت کاربر به دو روش صورت می‌گیرد:

- استفاده از کلمهٔ عبور
- رمزهایی که با کلید عمومی Client ها انجام می‌شوند.

قرارداد SSH2 معماری لایه‌ای جدیدی دارد. شکل ۴-۱۲ این معماری را نشان می‌دهد. در ادامه، عمل هر یک از لایه‌ها توضیح داده شده‌است.



شکل ۴- ۱۲ معماری SSH2

- لایهٔ انتقال (Transport Layer) تصدیق هویت سرور و محرمانگی داده و صحت آنها را به‌عهده می‌گیرد که در آن از الگوریتم‌های بلوکی 3DES، Blowfish، Twofish، IDEA، و CAST128 استفاده می‌شود.
- لایهٔ تصدیق کاربر (User Authentication Layer) که تصدیق هویت کاربر را انجام می‌دهد.
- لایهٔ ارتباط (Connection Layer) امکان برقراری Login به‌صورت محاوره‌ای را برقرار می‌کند و اجازه برقراری چندین نشست را بر روی یک کانال می‌دهد.

به‌طور خلاصه SSH1 وSSH2 بخش‌های مختلف یک بسته را رمزنگاری می‌کنند و پیاده‌سازی‌های شبکه‌ای شبیه‌به‌هم ندارند.

۴-۲-۴-۶. پروتکل IPSec

IPSec امنیتی را در سطح IP تأمین می‌کند و جزء ضروری‌ترین پروتکل‌های VPN است. IPSec شامل توابعی است که سرویس‌های تصدیق هویت و رمزنگاری را برای بسته‌های IP تأمین می‌کند. به همین دلیل برای ساختن VPN بر روی

یک شبکهٔ غیرامن و برای امن‌نمودن بسته‌ها بین هردو کامپیوتر در یک شبکهٔ محلی بکار می‌رود. IPSec از دو پروتکل AH و ESP استفاده می‌کند؛ AH برای تصدیق هویت و ESP برای رمزنگاری. این قرارداد عملیاتش را در دو مد انتقال (Transport) و تونل (Tunnel) انجام می‌دهد.

برای پیکربندی IPSec الگوریتم‌های رمزنگاری زیر استفاده می‌شود:

- برای رمزنمودن داده‌ها، از DES ۵۶ بیتی استفاده می‌شود.
- برای چک صحت داده از الگوریتم‌های درهم‌سازی MD5 یا SHA استفاده می‌گردد.
- برای تصدیق صحت مبدأ فرستندهٔ اطلاعات، از الگوریتم RSA استفاده می‌شود.

۴-۳. رمزشکنی[1] و حملات[2] علیه سیستم‌های رمزنگاری

حملات علیه سیستم‌های رمزنگاری، روشهایی هستند که رمزشکن ممکن است به کار ببرد تا امنیت یک رمزکننده را بشکند یا به آن نفوذ کند. این روش‌ها الگوریتم نیستند؛ آن‌ها فقط معابری به عنوان مکان شروع برای ایجاد الگوریتم‌های مشخص هستند. به طور کلاسیک، حملات نه نامگذاری شده‌اند و نه دسته‌بندی؛ تنها گفته شده است: " Here is a cipher, and here is the attack". هرچندکه حملات به آرامی، به حملات دارای نام، تبدیل شده‌اند اما هنوز طبقه‌بندی سراسری برای آنها وجود ندارد. در حال حاضر، حملات در درجهٔ اول با میزان اطلاعات در دسترس حمله‌کننده یا محدودیت‌های روی حمله و سپس با استراتژی‌هایی که از اطلاعات در دسترس استفاده می‌کنند، دسته‌بندی می‌شوند.

به طور کلی، نه تنها رمزکننده‌ها، بلکه توابع درهم‌سازی رمزنگاری نیز می‌توانند با استراتژی‌های خیلی متفاوت مورد حمله واقع شوند. رمزشکنی هنر رمزگشایی ارتباطات رمزشده، بدون داشتن کلیدهای مناسب می‌باشد. تکنیک‌های رمزشکنی زیادی موجودند؛ بعضی از مهمترین آنها برای یک پیاده‌ساز سیستم در ادامه تشریح شده‌اند.

۴-۳-۱. حملات مهم

۴-۳-۱-۱. حملهٔ Ciphertext-only

این وضعیتی است که حمله‌کننده چیزی دربارهٔ محتویات پیام نمی‌داند و باید فقط از Ciphertext به آن پی ببرد. در عمل، ممکن است که دربارهٔ Plaintext بتوان حدس‌هایی زد، چرا که انواع زیادی از پیام‌ها دارای سرآیند[3] با شکل ثابتی هستند. هنوز هم نامه‌های معمولی و اسناد به طریق خیلی قابل پیش‌بینی شروع می‌شوند. برای مثال، حملات کلاسیک زیادی از تحلیل فرکانسی Ciphertext استفاده می‌کنند، هر چند که، این روش در برابر رمزکننده‌های پیشرفتهٔ خوب کارآمد نیست. سیستم‌های رمزنگاری پیشرفته در برابر حملات Ciphertext-only ضعیف نیستند، چراکه گاهی اوقات آنها با فرض اضافه‌شده‌ای که پیام حاوی بعضی خصوصیات آماری می‌باشد در نظر گرفته می‌شوند.

۴-۳-۱-۲. حملهٔ Known-Plaintext

در این وضعیت، حمله‌کننده می‌داند یا می‌تواند Plaintext را برای بعضی بخش‌های Ciphertext حدس بزند. کار رمزگشایی باقیماندهٔ بلوک‌های Ciphertext با استفاده از این اطلاعات صورت می‌گیرد. این ممکن است به وسیله تشخیص کلید مورد استفاده برای رمزکردن داده، یا از طریق تعدادی میان‌بر انجام شود. یکی از بهترین حملات شناخته‌شدهٔ مدرن Known-plaintext رمزشکنی خطی علیه رمزکننده‌های بلوکی می‌باشد.

[1] Cryptanalysis
[2] Attacks
[3] Header

۴-۳-۱-۳. حملهٔ Chosen-Plaintext

در این وضعیت، حمله‌کننده قادر به داشتن رمزشدهٔ هر متن دلخواه با کلید ناشناخته می‌باشد. عمل لازم، مشخص‌کردن کلید استفاده شده برای رمزکردن می‌باشد. یک مثال از این حمله "رمزشکنی تفاضلی"[1] است که می‌تواند علیه رمزکننده‌های بلوکی به کار گرفته شود (و در بعضی حالات علیه توابع درهم‌سازی نیز استفاده می‌شود). بعضی سیستم‌های رمزنگاری، به‌طورمشخص RSA، نسبت به حملات Chosen-Plaintext آسیب‌پذیر هستند. هنگامی‌که چنین الگوریتم‌هایی استفاده می‌شوند، در طراحی برنامهٔ کاربردی (یا قرارداد) باید دقت شود که یک حمله‌کننده به هیچ‌وجه رمزشدهٔ Plaintext منتخبش را نداشته باشد.

۴-۳-۱-۴. حملهٔ Man-in-the-middle

این حمله مربوط به ارتباطات رمزنگاری و قراردادهای مبادلهٔ کلید می‌باشد. ایده این است که هنگامی‌که دو طرف A و B در حال مبادلهٔ کلید برای ارتباط امن می‌باشند (مثلاً با استفاده از Diffie-Hellman)؛ دشمن خودش را روی خط ارتباطی بین A و B قرار می‌دهد. دشمن سپس سیگنال‌هایی را که A و B به یکدیگر می‌فرستند قطع می‌کند و یک مبادلهٔ کلید به صورت جداگانه با A و B انجام می‌دهد. A و B به کار خود خاتمه می‌دهند در حالیکه از دو کلید متفاوت استفاده می‌کنند که هر کدام نزد دشمن شناخته شده‌است. دشمن سپس می‌تواند هر ارتباطی از A را با کلیدی که با A مشترک است رمزگشایی کند و مکاتبه را با رمزکردن آن با کلیدی که با B به اشتراک گذاشته است، به B بفرستد. هر دوی A و B فکر خواهند کرد که آنها به صورت امن در حال مکاتبه هستند، اما درحقیقت دشمن همه چیز را در کنترل خود آورده است.

راه معمول برای جلوگیری از حملهٔ Man-in-the-middle، استفاده از یک سیستم رمزنگاری کلید عمومی با توانایی ارائه امضاهای دیجیتالی می‌باشد.

برای شروع، طرفین باید از قبل کلید عمومی یکدیگر را بدانند. بعد از این که رمز اشتراکی تولید شد، طرفین امضاهای دیجیتالی آن را به یکدیگر می‌فرستند. Man-in-the-middle می‌تواند برای جعل این امضاها تلاش کند، اما شکست می‌خورد زیرا او نمی‌تواند امضاها را جعل کند.

این راه‌حل در ظهور راهی برای توزیع امن کلیدهای عمومی کفایت می‌کند. یک چنین راهی یک "سلسله‌مراتب گواهی"[2] نظیر X.509 می‌باشد. برای مثال در IPSec از این روش استفاده می‌شود.

[1] Differential cryptanalysis
[2] Certificate hierarchy

۴-۳-۱-۵. تشابه

تشابه[1] بین کلید محرمانه و خروجی سیستم رمزنگاری منبع اصلی اطلاعات برای رمزشکن می‌باشد. در ساده‌ترین حالت، اطلاعات مربوط به کلید محرمانه به‌طور مستقیم به‌وسیلهٔ سیستم رمزنگاری افشا می‌شود (نشت می‌کند). حالات خیلی پیچیده‌تر به مطالعهٔ تشابه (اساساً، هررابطه‌ای به تنهایی بر پایهٔ شانس مورد انتظار نخواهد بود) بین اطلاعات مشاهده شده (یا شمرده شده) درمورد سیستم رمزنگاری و اطلاعات کلید حدس زده شده نیاز دارد. به عنوان مثال، در حملات خطی (تفاضلی) علیه رمزکننده‌های بلوکی، رمزشکن، Known Plaintext (Chosen Plaintext) و Ciphertext مشاهده‌شده را مطالعه می‌کند. حدس‌زدن بعضی از بیت‌های کلید سیستم رمزنگاری توسط تحلیل‌گر به‌وسیلهٔ تشابه بین Plaintext و Ciphertext هر جا که وی به درستی حدس زده‌است؛ صورت می‌گیرد. این عمل می‌تواند تکرار شود، و نیاز به اصلاحات زیادی دارد.

رمزشکنی تفاضلی مطرح‌شده بوسیلهٔ Eli Biham و Adi Shamir در اواخر دههٔ ۱۹۸۰، اولین حمله‌ای بود که این ایده را به طورکامل علیه رمزکننده‌های بلوکی (به ویژه DES) به کارگرفت. بعداً Mitsuru Matsui رمزشکنی خطی را ارائه کرد که در برابر DES خیلی مؤثرتر بود. اخیراً، حملات جدیدی با استفاده از ایده‌های مشابه ارائه شده‌اند. شاید بهترین معرفی این مورد (این نوع) اقدامات EUROCRYPT و CRYPTO در سراسر دههٔ ۱۹۹۰ باشد. در آنجا می‌توان توضیحات Mitsuru Matsui دربارهٔ رمزشکنی خطی DES و ایده‌های"تفاضلات بریده‌شده"[2] به‌وسیلهٔ Lars Knudsen (به عنوان مثال، رمزشکنی IDEA) پیدا نمود. کتاب Eli Biham و Adi Shamir دربارهٔ رمزشکنی تفاضلی DES کار کلاسیکی دربارهٔ این موضوع بود.

ایدهٔ تشابه، در رمزنگاری، یک ایدهٔ اساسی است و چندین محقق تلاش کرده‌اند تا سیستم‌های رمزنگاری را ایجاد کنند که به طور اثبات‌پذیر علیه حملات این‌چنینی امن باشند. به عنوان مثال، Knudsen و Nyberg درمورد امنیت اثبات‌پذیر در برابر رمزشکنی تفاضلی مطالعه کرده‌اند.

۴-۳-۱-۶. حمله علیه یا با استفاده از سخت‌افزار زیر لایه

در چند سال اخیر از آنجایی‌که"وسایل رمزنگاری متحرک"[3] کوچک و کوچکتر، مورد استفادهٔ وسیع قرار گرفته‌اند، یک دستهٔ جدید از حملات ظهور پیدا کرده‌اند که مستقیماً پیاده‌سازی سخت‌افزاری سیستم رمزنگاری را هدف‌گیری می‌کنند. حملات از داده‌هایی که از اندازه‌گیری‌های خیلی ظریف وسیلهٔ مزبور در زمانهای مشخص، مثلاً در زمان رمزکردن بدست می‌آید استفاده می‌کنند و اطلاعات کلید را از این اندازه‌گیری‌ها محاسبه می‌کنند. بنابراین، ایده‌های اصلی تا حد زیادی به آن‌هایی که در دیگر حملات تشابه استفاده می‌شوند، وابسته است. برای مثال، حمله‌کننده تعدادی از بیت‌های کلید را حدس

[1] Correlation
[2] Truncated differentials
[3] Mobile crypto devices

می‌زند و تلاش می‌کند تا درستی حدس را با مطالعهٔ تشابه اندازه‌گیری‌هایش بررسی کند. چندین حمله نظیر استفاده از زمانبندی‌های دقیق وسیله، اندازه‌گیری‌های ظریف توان مصرفی و الگوهای تشعشع پیشنهاد شده‌اند. این اندازه‌گیری‌ها می‌تواند برای بدست‌آوردن کلید محرمانه یا دیگر انواع اطلاعات ذخیره شده روی وسیله به‌کار رود.

این حمله به طورکلی مستقل از الگوریتم‌های رمزنگاری مورد استفاده است و می‌تواند علیه هر وسیله‌ای که مشخصاً در برابر آن محافظت نشده است به کار رود.

اطلاعات بیشتر در برابر "آنالیز توان تفاضلی"[1] در آدرس زیر موجود است:

http://www.cryptography.com/dpa/

۴-۳-۱-۷. نقص‌ها در سیستم‌های رمزنگاری

"نقص‌ها در سیستم‌های رمزنگاری"[2] می‌توانند منجر به رمزشکنی و حتی افشای کلید محرمانه گردند. تمایل به وسایل رمزنگاری منجر به این کشف شد که بعضی الگوریتم‌ها با ایجاد نقصهای کوچک در محاسبات داخلی خیلی بد رفتار می‌کنند. به عنوان مثال، در پیاده‌سازی معمول RSA (بدون در نظرگرفتن بعضی ملاحظات)، اعمال مربوط به کلید خصوصی، خیلی در برابر این دسته از حملات آسیب‌پذیر می‌باشد. نشان داده شده‌است که با ایجاد یک بیت خطا در مکان مناسب می‌توان تجزیهٔ پیمانه را آشکار کرد (بدین‌وسیله کلید خصوصی هم آشکار می‌شود).

ایده‌های مشابهی در مورد طیف وسیعی از الگوریتم‌ها و وسیله‌ها به کار رفته‌اند. بدین ترتیب لازم است وسایل رمزنگاری که طراحی شده‌اند خیلی در برابر نقص (Fault) مقاوم باشند (و همچنین در برابر ایجاد نقص از روی بدخواهی به وسیلهٔ رمزشکن‌ها نیز مقاوم باشند).

۴-۳-۱-۸. "محاسبات کوانتومی"[3]

مقالهٔ Peter Shor دربارهٔ الگوریتم‌های تجزیه و لگاریتم‌گسسته با زمان چندجمله‌ای توسط کامپیوترهای کوانتومی، تمایل رو به تزایدی در محاسبات کوانتومی را ایجاد کرده‌است. محاسبهٔ کوانتومی یک زمینهٔ جدید از تحقیقات است که از مکانیک کوانتوم برای ساختن کامپیوترهایی که در تئوری خیلی قوی‌تر از کامپیوترهای سریال پیشرفته می‌باشند؛ استفاده می‌کند. این توانایی، از موازی‌سازی‌های ذاتی مکانیک کوانتوم حاصل می‌شود. بنابراین به جای انجام وظیفه‌ها یکی در هر زمان، آن طوری که ماشین‌های سریال انجام می‌دهند؛ کامپیوترهای کوانتوم می‌توانند همهٔ آنها را به‌یکباره انجام دهند. لذا امید است که به‌وسیلهٔ کامپیوترهای کوانتومی بتوان مسائلی را حل نمود که حلشان به وسیلهٔ ماشین‌های سریال غیرممکن است. نتایج Shor به‌طور ضمنی بیان می‌کند که اگر کامپیوترهای کوانتومی بتوانند به‌طور مؤثری پیاده‌سازی شوند؛ غالب روشهای رمزنگاری کلید عمومی به تاریخ خواهند پیوست.

[1] Differential power analysis
[2] Faults in cryptosystems
[3] Quantum computing

محاسبات کوانتومی پیشرفتهٔ فعلی نگران‌کننده به‌نظر نمی‌رسند؛ چراکه فقط ماشین‌های خیلی کوچکی پیاده‌سازی شده‌اند. نظریهٔ محاسبات کوانتومی کارآیی بهتری نسبت به کامپیوترهای سریال ارائه می‌دهد، اما اینکه، آیا می‌تواند درعمل تحقق یابد، یک سؤال بدون جواب است.

مکانیک کوانتوم همچنین منبعی برای روش‌های جدید رمزکردن داده و مکاتبهٔ امن با پتانسیل ارائهٔ امنیت غیرقابل شکست می‌باشد، این منبع، همان زمینهٔ رمزنگاری کوانتومی می‌باشد. بر خلاف محاسبهٔ کوانتومی، پیاده‌سازی‌های آزمایشی موفق زیادی از رمزنگاری کوانتومی پیش از این به انجام رسیده است. به هر حال، هنوز تحقق آن در کاربردهای تجاری، قدری دور از ذهن به نظر می‌رسد.

۴-۳-۱-۹. رمزنگاری DNA

Leonard Adleman (یکی از ارائه‌کنندگان RSA) استفاده از DNA را به عنوان کامپیوتر پیشنهاد داده است. مولکول‌های DNA می‌توانند به عنوان یک کامپیوتر خیلی بزرگ با توانایی انجام اعمال موازی در نظر گرفته شوند. این طبیعت موازی می‌تواند به کامپیوترهای DNA توانایی افزایش سرعت نمایی را در برابر کامپیوترهای سریال مدرن بدهد. متأسفانه مشکلاتی در مورد این کامپیوترهای DNA وجود دارند، که یکی می‌تواند این مورد باشد که افزایش سرعت نمایی، همچنین رشد نمایی حجم مواد مورد نیاز را لازم دارد. بنابراین، در عمل، بر روی کارآیی کامپیوترهای DNA، محدویت وجود دارد. همچنین ساخت چنین کامپوتری، خیلی ساده نیست.

۴-۳-۱-۱۰. حملهٔ Known-Ciphertext

در این حمله، رمزشکن یک نسخهٔ رمزشده از یک پیام (یا چند پیام رمزشده با یک الگوریتم) را دارد. یک نسخهٔ معمولی از حملهٔ Known-Ciphertext حملهٔ brute-fore می‌باشد که هر کلید ممکن برای رمزگشایی Ciphertext به کار می‌رود.

۴-۳-۱-۱۱. حملهٔ Related-Key

در این حمله، رمزشکن از دانش رابطهٔ بین کلیدهای مختلف و خروجی‌های مربوطهٔ آنها برخوردار است و از آنها برای حدس‌زدن کلیدی که برای رمزکردن چیزی به کار رفته‌است، استفاده می‌کند.

حملات رمزنگاری و تکنیک‌های رمزشکنی زیادی وجود دارند. به هر حال، موارد ذکرشده، مهمترین آنها برای یک طراح سیستم رمزنگاری می‌باشد.

۴-۳-۲. دسته‌بندی حملات و رمزشکنی

از آنجاکه دسته‌بندی مشخصی وجود ندارد تنها به ذکر عناوین اکتفا می‌شود (در مواردی که قبلاً صحبت نشده است توضیحاتی داده خواهد شد).

۴-۳-۲-۱. "محدویت‌های آگاهی"[1]

میزان اطلاعات در دسترس لزوماً استراتژی‌های حملهٔ دشمن را تحمیل می‌کند.

- Ciphertext Only
- Known Plaintext
- Defined Plaintext (همچنین Chosen Ciphertext و Adaptive Chosen Ciphertext). در این حمله، دشمن می‌تواند پیام‌های دلخواهی را برای رمزشدن ارائه کند و Ciphertextهای حاصل را جمع‌آوری کند.
- Chosen Key (یا Related-Key)
- Timing: یکی از انواع حملات سخت‌افزاری می‌باشد که قبلاً در مورد آن صحبت شده است.
- Fault Analysis: دشمن می‌تواند نقص‌های تصادفی را به ماشین‌آلات رمزکننده القاء کند، و از آن‌ها برای کشف کلید استفاده کند.
- Man-in-the-Middle

۴-۳-۲-۲. "استراتژی‌های حمله"[2]

یک حمله که با تلاش کمتری نسبت به جستجوی Brute-Force موفق می‌شود، یک شکستن[3] نامیده می‌شود. یک شکستن "آکادمیک" ("نظری") ممکن است مقادیر زیادی از داده یا منابع را درگیر کند، اما اگر بازهم ساده‌تر از Brute-Force باشد، شکستن نامیده می‌شود. (بنابراین ممکن است یک رمزکنندهٔ "شکسته‌شده" قوی‌تر از یک رمزکنندهٔ با کلید کوتاه‌تر باشد).

- Brute-Force (همچنین "جستجوی کلید سراسری"[4])
- Codebook (مفهوم "شکستن کد"[5] کلاسیک). در این حمله، دشمن سعی می‌کند که یک لیست یا یک کتاب از تمام تبدیلات ممکن بین Plaintext و Ciphertext با یک کلید مجزا ایجاد کند. یک راه مقابله داشتن اندازهٔ بلوک بزرگ می‌باشد.
- شکستن رمز تفاضلی[6]: در این استراتژی هدف، یافتن تشابه آماری بین مقادیر کلید و تبدیلات رمزکننده می‌باشد.

[1] Informational constraints
[2] Attack strategies
[3] Break
[4] Exhaustive key search
[5] Codebreaking
[6] Differential cryptanalysis

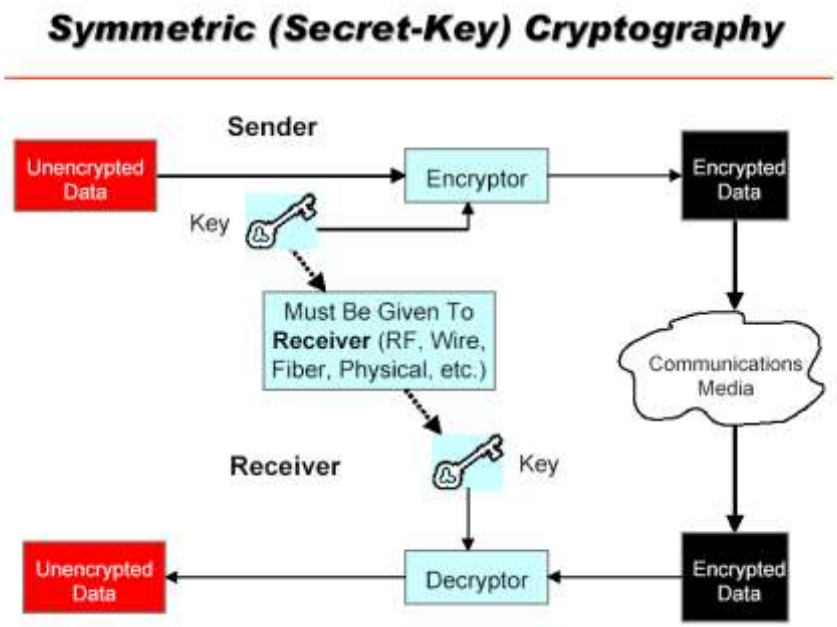
- شکستن‌رمز خطی[1]: هدف، یافتن یک تخمین خطی برای S-box های موجود در رمزکننده و استفاده از آن برای یافتن کلید می‌باشد.
- Meet-in-the-Middle
- Key Schedule: انتخاب کلیدهایی که تأثیرات معلومی را در مراحل مختلف رمزکردن به‌جای می‌گذارند.
- Birthday (معمولاً یک حمله علیه hash): با استفاده از پارادوکس روز تولد انجام می‌شود، ایده این است که یافتن دو مقدار که با هم تطبیق داشته باشند از یافتن تطبیق با یک مقدار مشخص، ساده‌تر است. پارادوکس اصلی بدین صورت می‌باشد که در یک کلاس درس تنها با ۲۳ دانش‌آموز با احتمال ۵۰ درصد، حداقل ۲ نفر متولد یک روز هستند.
- Correlation
- Dictionary: از یک لیست با کلیدهای خیلی محتمل، سعی در یافتن کلید مورد نظر صورت می‌گیرد (درواقع یک راه برای بهبود Brute-force می‌باشد). معمولاً برای یافتن یک کلمهٔ عبور استفاده می‌شود بدین نحوکه یک فرهنگ از کلمه‌های عبور متداول، ایجاد شده و روی آن Brute-Force انجام می‌شود.
- Replay: تعدادی از بلوک‌ها یا پیام‌های Ciphertext ضبط و ذخیره می‌گردد و سپس در زمان مناسب دوباره فرستاده می‌شود.

[1] Linear cryptanalysis

۴-۴. رمزکننده‌های بلوکی

یک دستهٔ بسیار مهم از الگوریتم‌های رمزنگاری، "رمزکننده‌های بلوکی"[1] می‌باشند. این رمزکننده‌ها در هر زمان، تعدادی بیت می‌گیرند (نوعاً ۶۴ بیت در رمزکننده‌های پیشرفته) وآنها را به عنوان یک واحد مستقل رمز می‌کنند.

در شکل ۴-۱۳ چگونگی استفاده از رمز نگاری متقارن مشاهده می‌شود.



شکل ۴- ۱۳ نحوهٔ عمل الگوریتم‌های متقارن

جنبه‌های مثبت ومنفی رمزنگاری کلید متقارن در زیر خلاصه شده‌اند:

- مزایا:
  - سریع می‌باشند.
  - به سادگی در سخت افزار پیاده‌سازی می‌شوند.
  - به طور گسترده استفاده می‌شوند.
- معایب:
  - برای هر فرد تازه وارد باید یک کلید جدید تولید و نگهداری شود تا بتواند به مبادلهٔ اطلاعات بپردازد (n نفر-[2] n)/2 کلید نیاز دارند).

[1] Block ciphers

- کلید محرمانه باید از طریق یک کانال مطمئن (امن) مبادله شود.
- باید کلید با طول ثابت استفاده شود.
- در صورت استفاده از الگوریتم ضعیف قابل شنود است.
- تلاش و زحمت بیشتری برای تأیید فرستنده نیاز است.
- به یک منطق ادارهٔ کلید (key management) نیاز است.

الگوریتم DES[1] الگوریتم متقارنی است که خیلی زیاد مورد مطالعه قرارگرفته و در بین الگوریتم‌های متقارن بیشترین کاربرد را داشته است. البته AES در آینده‌ای نزدیک ممکن است به عنوان الگوریتم رمزکنندهٔ پرکاربرد، جایگزین آن شود. به دلیل اهمیت وگستردگی استفاده‌ای‌که DES دارد در بررسی الگوریتم‌های متقارن در ابتدا به سراغ این الگوریتم می‌رویم.

۴-۴-۱. الگوریتم رمزنگاری DES

برای سادگی، اطلاعات در چند بخش تقسیم شده‌اند تا قضاوت بهتری در مورد این الگوریتم صورت گیرد.

۴-۴-۱-۱. تاریخچه

الگوریتم DES در اواسط دههٔ ۷۰ در شرکت IBM طراحی شد. در سال ۱۹۷۶ به عنوان الگوریتم رمزنگاری استاندارد توسط انستیتوی ملی استانداردها و فن آوری آمریکا[2] برای داده‌های غیرسری انتخاب شد. یعنی به عنوان استاندارد پردازش اطلاعات فدرال[3] وانستیتوی استانداردهای ملی امریکا[4] تحت عنوان X3.92 انتخاب شد. پایان سلطنت طولانی آن به عنوان الگوریتم رسمی NIST در سال ۱۹۹۷ اتفاق افتاد. جالب توجه است که آژانس امنیت ملی[5] آمریکا به طور غیرمستقیم توصیه‌هایی برای طراحی S-boxهای این الگوریتم کرده بود که نقطهٔ قوت این الگوریتم و نشان دهندهٔ میزان آگاهی آژانس امنیت ملی در بدو طراحی DES می‌باشد.

۴-۴-۱-۲. توضیح مختصر الگوریتم

DES یک رمزکنندهٔ بلوکی است که داده‌ها را در بلوکهای ۶۴ بیتی رمز می‌کند. یک بلوک ۶۴ بیتی از plaintext در یک طرف الگوریتم وارد می‌شود و یک بلوک ۶۴ بیتی از ciphertext از طرف دیگر خارج می‌شود. DES یک الگوریتم

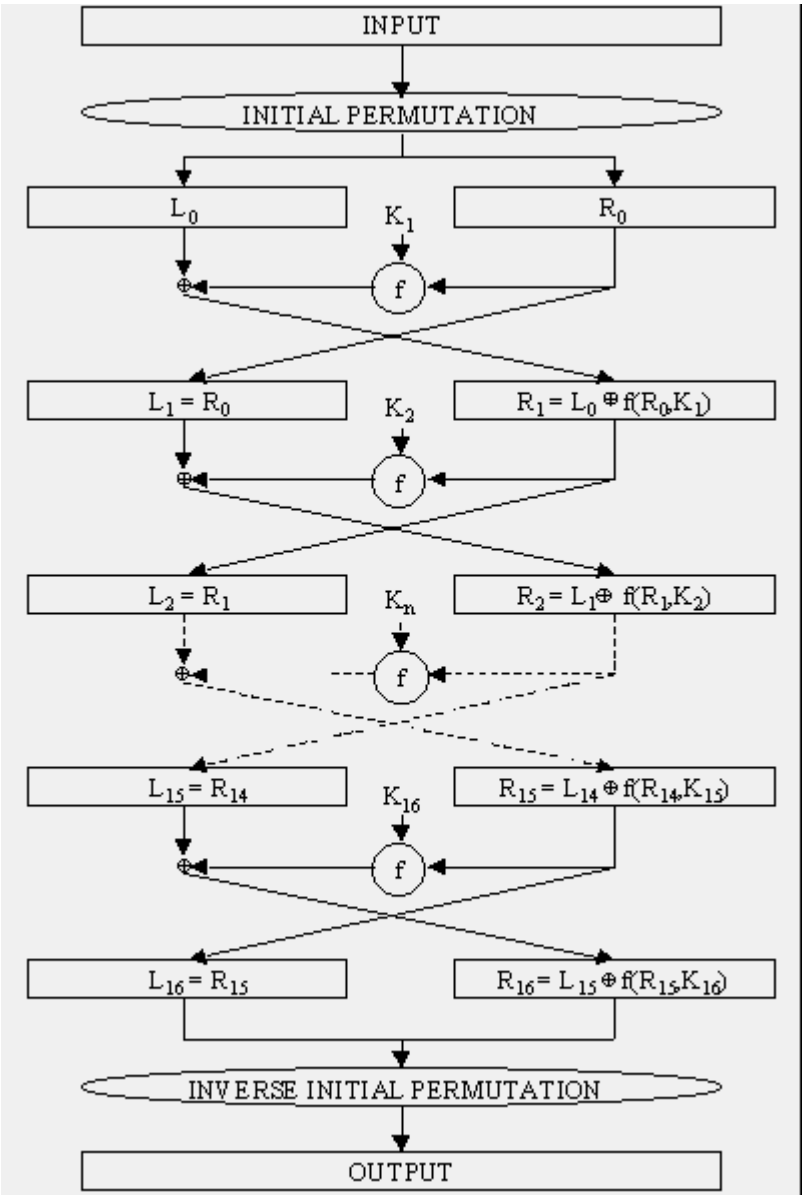
[1] Data Encryption Standard
[2] National Institute of Standards and Technology (NIST)
[3] Federal Information Processing Standard 46 (FIPS 46-2)
[4] ANSI
[5] National Security Agency (NSA)

متقارن است؛ لذا در آن، **یک** الگوریتم و **یک** کلید برای رمزکردن و رمزگشایی مورد استفاده قرار می‌گیرد (بجز تفاوتهای کوچک در زمانبندی کلید).

طول کلید مورد استفاده در این الگوریتم، ۵۶ بیت می‌باشد (عموماً بیان می‌شود که کلید یک عدد ۶۴ بیتی است، اما هشت بیت برای بررسی توازن استفاده می‌شود و ازآنها صرفنظر می‌شود). کلید، هر عدد ۵۶ بیتی می‌تواند باشد و در هر زمان می‌تواند تغییرکند. تعدادی از اعداد به عنوان کلیدهای ضعیف شناخته شده‌اند، اما به راحتی می‌توان از آنها جلوگیری‌کرد. تمام امنیت وابسته به کلید می‌باشد.

الگوریتم DES روی بلوکهای ۶۴ بیتی، ۱۶ مرحله (round) از جایگشتها[1]، معاوضه‌ها[2] وجانشینی‌ها[3] را همانطور که در شکل ۴-۱۴ نشان داده شده است، انجام می‌دهد. این استاندارد همچنین جدولهایی را که توصیف‌کنندهٔ تمام اعمال انتخاب، جایگشت، و بسط انجام‌گرفته درطی الگوریتم می‌باشند، شامل می‌شود. این مفاهیم الگوریتم سری (Secret) نیستند.

[1] Permutations
[2] Swaps
[3] Substitutions



شکل ۴-۱۴ چگونگی عمل الگوریتم DES

قدمهای اصلی DES این چنین هستند:

۱. بلوک ۶۴ بیتی که می‌خواهد رمزشود، دستخوش یک جایگشت اولیه[1] می‌شود؛ یعنی هر بیت به یک مکان جدید منتقل می‌شود. به عنوان مثال، بیتهای یکم، دوم، و سوم به ترتیب به مکانهای، ۵۸ اُم ،۵۰ اُم ، و ۴۲ اُم منتقل می‌شوند.

۲. ورودی ۶۴ بیتی دچار جایگشت شده به دو بلوک ۳۲ بیتی به نامهای چپ و راست، به ترتیب تقسیم می‌شود. مقادیر اولیهٔ بلوکهای چپ و راست با L_0 وR_0 مشخص می‌شوند.

۳. سپس ۱۶ بار[1] عملیات روی بلوکهای L و R انجام می‌شود. در خلال هر تکرار (n از۱ تا ۱۶ تغییر می‌کند)، فرمولهای زیر اعمال می‌شوند:

[1] Initial Permutation (IP)

$L_n = R_{n-1}$

$R_n = L_{n-1}$ XOR $f(R_{n-1}, K_n)$

در هر گام داده‌شده از فرآیند، مقدار جدید بلوک L فقط از مقدار قبلی بلوک R گرفته‌می‌شود. بلوک جدید R ازمحاسبهٔ XOR بلوک قبلی L با نتایج اعمال تابع رمزکنندهٔ f بر بلوک قبلی R و K_n محاسبه می‌شود. (K_n یک مقدار ۴۸ بیتی مشتق‌شده از کلید ۶۴ بیتی DES می‌باشد. هر مرحله، از ۴۸ بیت متفاوتی برحسب الگوریتم زمانبندی استاندارد کلید استفاده می‌کند.)

تابع رمزکنندهٔ f ، مقدار ۳۲ بیتی بلوک R و زیرکلید ۴۸ بیتی را به روش زیر با هم ترکیب می‌کند. نخست ۳۲ بیت بلوک R بوسیلهٔ تابع بسط E به ۴۸ بیت بسط داده می‌شود؛ ۱۶ بیت اضافه با تکرار ۱۶ بیت از مکانهای از قبل تعریف شده مهیا می‌شوند.بلوک R بسط داده شده با زیرکلید ۴۸ بیتی OR می‌شود. حاصل، به ۸ بلوک ۶ بیتی تقسیم می‌شود. اینها به ورودی هشت S-box[۲] به نامهای S_1 تا S_8 اعمال می‌شوند. هر ورودی ۶ بیتی یک S-box با استفاده از یک جدول Lookup ، ۴ بیت خروجی را نتیجه می‌دهد. سپس خروجی ۳۲ بیتی مجموعهٔ S-boxها بوسیلهٔ تابع جایگشت P دوباره مرتب می‌شود.

۴. نتایج، از دور نهایی DES - یعنی L_{16} و R_{16} - به هدف یک مقدار ۶۴ بیتی با هم ترکیب می‌شوند وبه معکوس جایگشت اولیه (IP^{-1}) وارد می‌شوند. در این گام، بیتها به مکانهای اصلی‌شان برگردانده می‌شوند. بنابراین بیتهای ۵۸ اُم، ۵۰ اُم، ۴۲ اُم، به عنوان مثال، به مکانهای یکم، دوم، و سوم بازگردانده می‌شوند. خروجی IP^{-1} ، ۶۴ بیت متن رمز شده می‌باشد.

۴-۴-۱-۳. امنیت DES

افراد مختلف پرسشهای زیادی در مورد امنیت DES مطرح کرده‌اند. تفکر زیادی روی طول کلید، تعداد تکرارها، طراحی S-boxها صورت‌گرفته‌است. S -boxها مخصوصاً مرموز بودند چراکه بدون هیچ دلیل آشکار همهٔ آنها باید ثابت می‌بودند. اگرچه که IBM اطمینان دادکه کارهای درونی این پروژه نتیجهٔ ۱۷ نفر- سال cryptanalysis شدید بوده است، بعضی ترسیدند که آژانس امنیت ملی (NSA) منافذی در الگوریتم جاسازی کرده‌است تا ابزار ساده‌ای را برای رمزگشایی همهٔ پیامها در اختیار داشته باشد.

از آنجائیکه کلید اولیه برای استخراج زیرکلید برای هر مرحله تغییر می‌یابد، کلیدهای اولیهٔ مشخصی کلیدهای ضعیفی هستند (مثلاً اگر همهٔ بیتها ۰ یا ۱ باشند).

[1] Round
[2] Selection-box or Substitution-box

ارائهٔ اصلی IBM به NIST یک کلید ۱۱۲ بیتی داشت و تا زمانی که DES استاندارد شد، به ۵۶ بیت کاهش پیداکرد. این کاهش اندازهٔ کلید بیشتر به خاطر پیاده‌سازی آن روی **یک** تراشه بود. رمزنگاران زیادی برای کلید بزرگتر استدلال نمودند. در واقع پیش‌بینی می‌شد که کلید ۵۶ بیتی DES برای مقاومت در برابرحملهٔ brute-force (امتحان نمودن تمام کلیدهای ممکن) از طرف کامپیوترهای پیشرفته، خیلی‌کم باشد. قانون مور[1] را بخاطر بیاورید که بیان می‌کرد قدرت کامپیوترها در هر ۱۸ ماه دو برابر می‌شود. در چنین حالتی حدود هر ۱۸ ماه باید به طول کلید یک بیت اضافه نمود تا امنیت قابل اطمینانی را در برابر این نوع حمله بدست آورد.

در سال ۱۹۹۸، EFF[2] ساخت یک سخت افزار را اعلام‌کرد که می‌توانست یک کلید DES را به صورت brute-force به طور متوسط در ۴/۵ (چهار و نیم) روز بیابد. این تراشه که Deep Crack نام داشت، می‌توانست ۹۰ میلیارد کلید را در هر ثانیه بررسی‌کند. سر انجام در سال ۱۹۹۹ با استفاده از Deep Crack و همکاری distributed.net در کمتر از یک روز موفق به شکستن آن شدند.

۴-۱-۴-۴. پیاده سازی

از اهداف طراحی DES، پیاده سازی آن در سخت‌افزار برای سرعت زیاد بود. به همین دلیل طول کلید را تا ۵۶ بیت کاهش دادند. به هر حال پیاده سازیهای گوناگونی چه سخت‌افزاری و چه نرم‌افزاری از این الگوریتم صورت‌گرفته است که بسته به نوع نیاز می‌توان ازآنها استفاده نمود. مثلاً در سال ۱۹۹۶ شرکت DEC، سریعترین تراشهٔ DES تا آن زمان را تولید نمود که با سرعت ۱ گیگابیت برثانیه یا ۱۶/۸ میلیون بلوک درثانیه می‌توانست داده‌ها را رمز و رمزگشایی‌کند.

۴-۱-۴-۵. گونه‌های مختلف DES

برای اصلاح نواقص DES، هر ازگاهی نوعی جدید از آن طراحی می‌شد که از آن جمله می‌توان به Multiple DES (معروفترین آن Triple-DES می‌باشد)، DES with Independent Subkeys ، DESX ، CRYPT(3) (نوعی از DES مخصوص سیستمهای یونیکس)، GDES ، DES with Alternative S-box ،RDES ،S^n DES ،DES with Key-Dependent S-boxes ،Biham-DES و... اشاره کرد. ثابت شده‌است که بعضی از این گونه‌ها در برابر بعضی حمله‌های خاص ویا روشهای مدرن Cryptanalysis از DES اصلی ضعیف‌تر می‌باشند.

- *3DES*

بر پایهٔ استفاده از ۳ بار DES (به طور معمول دریک توالی encryption-decryption-encryption با ۲ یا ۳ کلید مختلف و نامربوط) بنا نهاده شده است و بدین ترتیب طول مؤثر کلید تا حد زیادی افزایش می‌یابد. در دنیای بانکداری نوعاً از یک کلید ۱۱۲ بیتی استفاده می‌شود. جهان اینترنت یک کلید ۱۶۸ بیتی را استفاده می‌کند اما ثابت شده است که این کلید امنیت بیشتری نسبت به کلید ۱۱۲ بیتی ارائه نمی‌دهد. این الگوریتم هنگامی که در نرم افزار پیاده سازی می‌شود، خیلی کند است.

[1] Moore
[2] Electronic Frontier Foundation

لازم به ذکراست که به دلیل اینکه DES یک گروه (Group) ریاضی نمی‌باشد، ترکیبات متوالی آن منجر به افزایش طول مؤثر کلید شده است. البته این عمل سرعت کار را پایین می‌آورد.

- *DESX*

در سال ۱۹۹۶ برای افزایش مقاومت DES در برابر حمله‌های brute-force، الگوریتم خیلی ساده‌ای پیشنهاد شد که بدون اضافه‌کردن در پیچیدگی محاسباتی، به طرز قابل توجهی چنین انتظاری را برآورده می‌کند. در DESX، ورودی plaintext قبل از رمز شدن با ۶۴ بیت اضافهٔ کلید XOR می‌شود و در خروجی نیز چنین عملی تکرار می‌شود. با اضافه کردن تنها دو عمل XOR، DESX یک طول کلید مؤثر ۱۲۰ بیتی دربرابر یک حملهٔ جستجوی کلید سراسری[1] پیدا می‌کند؛ هرچندکه این روش مصونیت بیشتری نسبت به گونه‌های دیگر در برابر حملات سطح بالا، نظیر شکستن رمز تفاضلی یا خطی ندارد.

۴-۴-۲. الگوریتم رمزنگاری AES

انتخاب یک الگوریتم مناسب برای رمزنگاری از مهمترین ملزومات یک سیستم امن می‌باشد. جالب توجه است که **امروزه، محرمانه بودن، معیار خوبی یک الگوریتم نمی‌باشد**. صرفنظر از تئوری ریاضی پشت یک الگوریتم، بهترین الگوریتمها آنهایی هستند که مشهورند و به‌خوبی مستندسازی شده‌اند، زیرا آنها به خوبی مورد مطالعه و آزمایش قرارگرفته‌اند! در حقیقت، ***زمان***، تنها آزمایش صحیح یک رمزنگاری خوب است؛ هر روش رمزنگاری که سالها مورد استفاده قرار می‌گیرد به احتمال قوی یک روش خوب می‌باشد و خریداران باید از هر محصولی که از یک روش رمزنگاری اختصاصی استفاده می‌کند پرهیز کنند چراکه محرمانگی الگوریتم مزیتی برای آن الگوریتم نمی‌باشد.

واضح است که باید سراغ الگوریتمهایی رفت که دارای ویژگیهای ذکرشده باشد. به همین دلیل در این گزارش به الگوریتم AES (Rijndael) پرداخته شده‌است چراکه توسط مراجع بزرگی به‌خوبی مورد بررسی قرار گرفته است. امید است که این الگوریتم تا مدتها مورد استفاده قرار گیرد (البته تا زمانی‌که شکسته نشود و الگوریتم بهتری نیزجایگزین آن نگردد!).

۴-۴-۲-۱. چگونگی انتخاب الگوریتم AES

جستجو برای جانشین DES در ژانویهٔ ۱۹۹۷ آغاز شد. زمانی که NIST اعلام کرد که به دنبال "استاندارد رمزنگاری پیشرفته"[2] (AES) می‌باشد. در سپتامبر همان سال آنها یک فراخوان رسمی برای الگوریتم منتشر کردند و در آگوست ۱۹۹۸ اعلام کردند که ۱۵ الگوریتم کاندید از جمع ۲۱ الگوریتم اولیه مورد توجه قرار گرفته‌اند (مرحلهٔ اول). در آوریل ۱۹۹۹، NIST اعلام کرد که ۱۵ الگوریتم به پنج الگوریتم نهایی کاهش پیدا کرده است (مرحلهٔ دوم) که عبارتند از:

[1] Brute-force
[2] Advanced Encryption Algorithm

- *MARS*[1] *از شرکت IBM*
- *RC6 توسط رونالدریوست*[2]
- *Rijndael از یک تیم بلژیکی*
- *Serpent مشترکاً توسط یک تیم از انگلیس، اسرائیل و نروژ.*
- *Twofish توسط بروس اشنیر*[3]

در اکتبر ۲۰۰۰، NIST انتخاب خود را اعلام کرد: Rijndael

نکتهٔ استثنایی و قابل توجه در این مورد این است که علاوه بر طبیعت بین‌المللی "رقابت"، همهٔ فرایند آشکار و روراست بوده‌است. NIST یک پایگاه وب خیلی خوب که به اطلاع‌رسانی کامل وفادار است را در این‌باره در URL زیر نگهداری می‌کند:

http://csrc.nist.gov/encryption/aes

در اکتبر ۲۰۰۰، NIST گزارشی مبنی بر ظهور استاندارد رمزنگاری پیشرفته (AES) منتشر کرد که پنج الگوریتم مرحلهٔ دوم را در چند مقوله مقایسه می‌کرد. جدول ۴-۳ امتیازات نسبی این پنج روش را خلاصه می‌کند (۱ = ضعیف،۳ = عالی).

جدول ۴-۳ مقایسهٔ الگوریتم‌های مرحلهٔ پایانی برای انتخاب AES

	الگوریتم				
مقوله	MARS	RC6	Rijndael	Serpent	Twofish
امنیت کلی	۳	۲	۲	۳	۳
پیاده سازی امنیت	۱	۱	۳	۳	۲
کارآیی نرم‌افزار	۲	۲	۳	۱	۱
کارآیی کارت هوشمند	۱	۱	۳	۳	۲
کارآیی سخت‌افزار	۱	۲	۳	۳	۲
کیفیت طراحی	۲	۱	۲	۱	۳

به همراه گزارش توصیه‌نامه‌ای هم پخش گردید که Rijndael به نام AES نامگذاری شود. در فوریهٔ ۲۰۰۱ NIST پیش‌نویس استاندارد پردازش اطلاعات فدرال (FIPS) مشخصات AES را برای بازبینی عمومی و تجدیدنظر کلی پخش کرد. AES حاوی یک *زیرمجموعه* از توانایی‌های Rijndael می‌باشد (به عنوان مثال، AES فقط اندازهٔ بلوک ۱۲۸ بیتی را پشتیبانی می‌کند و از اصطلاحات متفاوتی نسبت به اصطلاحات استفاده‌شده توسط ایجادکنندگان Rijndael استفاده می‌کند،

[1] Multiplication, Addition, Rotation, and Substitution
[2] Ronald Rivest
[3] Bruce Schneir

اما فهمیدن یکی از این دو مجموعهٔ اصطلاحات، برای فهمیدن دیگری کفایت می‌کند.) مهلت ۵۰ روزهٔ توضیحات و پیشنهادات در ۲۹ ماه می ۲۰۰۱ پایان یافت و "وزارت تجارت"[1] ایالات متحده، رسماً AES را در دسامبر ۲۰۰۱ انتخاب کرد و به عنوان FIPS PUB197 منتشر کرد.

قابل ذکر است که AES یک الگوریتم رمزنگاری برای استفادهٔ سازمان‌های دولتی ایالات متحده برای حفاظت اطلاعات حساس و غیرسری می‌باشد. NIST پیش‌بینی می‌کند که AES به صورت گسترده‌ای توسط سازمان‌ها، مؤسسه‌ها و افراد خارج از دولت آمریکا و در بعضی حالات-خارج از ایالات‌متحده- مورد استفاده قرار خواهد گرفت. "وزیر تجارت"[2] دولت آمریکا انتخاب AES به عنوان استاندارد رسمی دولت را از ۲۶ می ۲۰۰۲ قابل اجرا دانست.

از اهداف NIST حداقل نمودن اندازهٔ بلوک و کلید و "دانش فنی"[3] مورد نیاز در الگوریتم مورد نظر بوده‌است. الگوریتم Rijndael ترکیبی از خصوصیات امنیت، کارآیی، بهره‌وری، سادگی پیاده‌سازی و انعطاف‌پذیری را داراست.

به‌وجودآورندگان Rijndael دکتر Joan Daemen[4] از Proton World International و دکتر Vincent Rijmen[5] محقق فوق دکترا در دانشکدهٔ مهندسی برق دانشگاه Katholieke Universiteit Leaven می‌باشند که هر دو از اعضای خیلی فعال انجمن رمزنگاری بوده‌اند. تلفظ Rijndael توسط ارائه دهندگان آن پیشنهاد شده است که شبیه یکی از ترکیبات زیر باشد: "Reign Dahl"، "Rain Doll" و یا "Rhine Dahl".

۴-۴-۲-۲. توضیح الگوریتم

Rijndael یک رمزکنندهٔ بلوکی متقارن است که می‌تواند بر روی بلوک با طول متغیر با استفاده از کلیدهای با طول متغیرعمل کند. نسخهٔ دوم مشخصات که به NIST ارائه شد، بکارگیری کلید ۱۲۸، ۱۹۲ و یا ۲۵۶ بیتی را برای رمزکردن بلوکهای داده‌ای که ۱۲۸، ۱۹۲ و یا ۲۵۶ بیت طول دارند توصیف می‌کند؛ توجه شود که همهٔ ۹ ترکیب طول کلید و طول بلوک امکان‌پذیر می‌باشند. الگوریتم به نحوی نوشته شده است که طول بلوک و طول کلید به‌سادگی قابل بسط داده‌شدن به اندازهٔ مضاربی از ۳۲ بیت می‌باشند و بخصوص برای پیاده‌سازی مؤثر در سخت‌افزار یا نرم‌افزار روی محدوده‌ای از پردازنده‌ها طراحی شد. طراحی Rijndael شدیداً متأثر از رمزکنندهٔ بلوکی به نام Square بوده که توسط Daemen و Rijmen طراحی شده بود.

الگوریتم Rijndael یک رمزکنندهٔ بلوکی تکراری می‌باشد، بدین معنی که بلوک ورودی اولیه و کلید رمز تحت اثر مراحل متعددی از تبدیلات، قبل از تولید خروجی می‌گردند. هر نتیجهٔ میانی رمز کننده یک *حالت*[6] نامیده می‌شود.

[1] Department of Commerce
[2] Secretary of Commerce
[3] Intellectual Property (IP)
[4] Yo'-ahn Dah'-mun
[5] Rye'-mun
[6] State

برای سادگی توصیف، بلوک وکلید رمز غالباً به صورت آرایه‌ای از ستونها نشان داده می‌شوند که هر آرایه ۴ سطر دارد و هر ستون یک بایت مجرد (۸ بیت) را نمایش می‌دهد. در این صورت، تعداد ستون‌ها در یک آرایهٔ نشان‌دهندهٔ حالت یا "کلید رمز"[1] با تقسیم طول بلوک یا طول کلید بر ۳۲ می‌تواند محاسبه شود. یک آرایهٔ نشان دهندهٔ یک حالت، N_b ستون خواهد داشت که مقادیر N_b به ترتیب، ۴، ۶، ۸، متناظر با بلوک ۱۲۸، ۱۹۲، ۲۵۶ بیتی می‌باشد. به طور مشابه، یک آرایهٔ نشان‌دهندهٔ یک کلید رمز، N_k ستون خواهد داشت که مقادیر N_k به ترتیب۴، ۶، ۸ متناظر با کلید ۱۲۸، ۱۹۲، ۲۵۶ بیتی می‌باشد. یک مثال از یک *حالت* ۱۲۸ بیتی ($N_b = ۴$) و *کلید رمز* ۱۹۲ بیتی ($N_k = ۶$) در زیر نشان داده شده‌است (جدول‌های ۴-۴ و ۴-۵).

جدول ۴-۴ یک کلید نمونه

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$	$k_{0,4}$	$k_{0,5}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$	$k_{2,4}$	$k_{2,5}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$	$k_{3,4}$	$k_{3,5}$

جدول ۴-۵ یک حالت نمونه

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

تعداد مراحل تبدیل (N_r) در Rijndael، تابعی از طول بلوک و طول کلید می‌باشد و توسط جدول ۴-۶ تعیین می‌شود.

جدول ۴-۶ تعداد مراحل تبدیل

No. of Rounds Nr		Block Size		
		128 bits Nb = 4	192 bits Nb = 6	256 bits Nb = 8
Key Size	128 bits Nk = 4	10	12	14
	192 bits Nk = 6	12	12	14
	256 bits Nk = 8	14	14	14

در حال حاضر نسخهٔ AES، همهٔ ۹ ترکیب طول بلوک و کلید را پشتیبانی نمی‌کند؛ فقط زیر مجموعه‌ای که از اندازهٔ بلوک ۱۲۸ بیتی استفاده می‌کند، پشتیبانی می‌شود. NIST این گونه‌های پشتیبانی شده را AES-128، AES-192 و AES-256 می‌نامد که عدد به اندازهٔ کلید اشاره می‌کند. مقادیری از N_b، N_k و N_r که در AES پشتیبانی می‌شوند در جدول ۴-۷ ذکر شده‌اند.

جدول ۴-۷ مقادیر پشتیبانی‌شده در AES

	Parameters		
Variant	Nb	Nk	Nr
AES-128	4	4	10

[1] Cipher key

AES-192	4	6	12
AES-256	4	8	14

رمز کنندهٔ Rijndael /AES، خود سه مرحلهٔ عمل دارد:

- تبدیل AddRoundKey
- N_r-1 مرحله شامل:
  - تبدیل SubBytes
  - تبدیل ShiftRows
  - تبدیل MixColumns
  - تبدیل AddRoundKey
- یک مرحلهٔ نهایی شامل:
  - تبدیل SubBytes
  - تبدیل ShiftRows
  - تبدیل AddRoundKey

بندهای زیر اعمال نامبرده شده در بالا را شرح خواهند داد. اصطلاحات استفاده شده در زیر از توصیفات AES گرفته شده‌است هرچندکه اشاره‌ای به اصطلاحات Rijndael برای کامل بودن مطلب نیز شده است. آرایه‌های s و s´ به ترتیب به حالت قبل و بعد از تبدیل اشاره می‌کنند (Rijndael از a وb استفاده می‌کند). زیرنویس‌های i وj برای نشان دادن مکان‌های بایت در داخل آرایهٔ حالت (یا کلید رمز) استفاده می‌شوند.

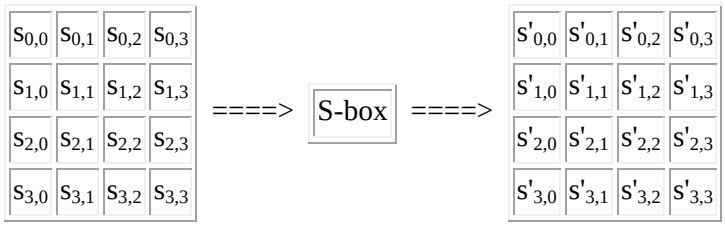
◆ تبدیل SubBytes

تبدیل جانشین‌کردن بایت‌ها (در Rijndael به نام ByteSub) روی هر کدام از بایت‌های حالت به طور مستقل عمل می‌کند و مقدار بایت را تغییر می‌دهد. یک S-box یا جدول جانشینی تبدیل را کنترل می‌کند. مشخصات تبدیل S-box علاوه بر جدول S-box در مشخصات AES ذکر شده‌اند؛ به عنوان مثال، یک مقدار بایت حالت ورودی ۱۰۷ (0x6b) با ۱۲۷ (0x7f) در حالت خروجی جایگزین خواهد شد و یک مقدار ورودی ۸ (0x08) با ۴۸ (0x30) عوض خواهد شد.

یک راه تصورکردن تبدیل SubBytes این است که یک بایت داده شده در حالت s به یک مقدار جدید در حالت ´s بر حسب S-box نگاشته می‌شود. بنابراین S-box تابعی روی یک بایت در حالت ورودی به صورت زیر می‌باشد:

$$S\text{-box}\ (s_{i,j}) = s'_{i,j}$$

نمایش جامع‌تر این تبدیل در شکل ۴-۱۵ نشان داده شده است.



شکل ۴- ۱۵ نمایش جامع‌تر تبدیل یک S-box

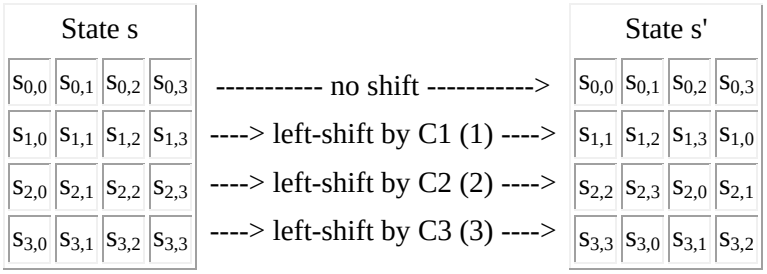
◆ تبدیل ShiftRows

تبدیل انتقال سطرها (به نام ShiftRow در Rijndael) به صورت دوره‌ای بایت‌های سه سطر پایین آرایهٔ حالت را تغییر مکان می‌دهد. سطرهای ۲، ۳، ۴ به صورت دوره‌ای به ترتیب به اندازهٔ C1، C2 و C3 بایت بر طبق جدول ۴-۸ تغییر مکان پیدا می کنند.

جدول ۴-۸ مقدار تغییر مکان (Shift) بر حسب مقدار N_b

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

نسخهٔ فعلی AES، فقط اندازهٔ بلوک ۱۲۸ بیتی (Nb = ۴) را اجازه می‌دهد. بنابراین C1 =۱، C2 =۲ و C3 =۳. شکل ۴-۱۶ تأثیر تبدیل ShiftRows را روی حالت s نشان می‌دهد.



شکل ۴-۱۶ نحوهٔ عمل تبدیل ShiftRows

◆ تبدیل MixColumns

تبدیل درهم‌کردن (مخلوط کردن) ستون‌ها (به نام MixColumn در Rijndael) از یک تابع ریاضی برای تبدیل مقادیر یک ستون داده شده در یک حالت استفاده می‌کند. عمل روی چهار مقدار در یک زمان همانند این است که آنها یک چهار جمله‌ای را نشان می‌دهند:

$$\text{MixColumns}\ (s_{i,c}) = s'_{i.c}$$

برای ۰ <= i <= ۳ و برای ستون c.

مکان ستون تغییر نمی‌کند و فقط مقادیر داخل ستون دستخوش تغییر می‌شوند.

◆ تولید Round Key و تبدیل AddRoundKey

کلید رمز AES می‌تواند ۱۲۸، ۱۹۲ یا ۲۵۶ بیتی باشد. کلید رمز برای تولید یک کلید متفاوت که در خلال هر مرحله از عمل رمزکردن روی بلوک اعمال می‌شود، مورد استفاده قرار می‌گیرد. این کلیدها، "کلیدهای مرحله‌ای"[1] نامیده می‌شوند و هر کدام هم‌طول بلوک خواهد بود، یعنی N_b کلمهٔ ۳۲ بیتی (کلمات با W مشخص خواهند شد). توصیف AES یک زمانبندی کلید تعریف می‌کند که بر اساس آن کلید رمز اصلی (به طول N_k کلمهٔ ۳۲ بیتی) برای ایجاد یک "کلید بسط‌یافته"[2] استفاده می‌شود. اندازهٔ کلید بسط‌یافته مساوی اندازهٔ بلوک ضربدر تعداد مراحل رمزکردن به اضافهٔ ۱ می‌باشد که N_r +۱ کلید متفاوت را در اختیار می‌گذارد. (توجه شود که N_r مرحلهٔ رمزکردن وجود دارند اما N_r +۱ تبدیل AddRoundKey انجام می‌شود).

در نظر بگیرید که AES از یک بلوک ۱۲۸ بیتی و ۱۰، ۱۲ یا ۱۴ مرحلهٔ تکراری بسته به طول کلید استفاده می‌کند. با یک کلید ۱۲۸ بیتی، به عنوان مثال، به ۱۴۰۸ بیت از "جنس کلید"[3] نیاز خواهد بود (۱۴۰۸ = ۱۱ * ۱۲۸)، با یک کلید بسط‌یافته به اندازهٔ ۴۴ کلمهٔ ۳۲ بیتی (۱۴۰۸ = ۳۲ * ۴۴). به همین ترتیب، یک کلید ۱۹۲ بیتی به ۱۶۶۴ بیت از جنس‌کلید نیاز خواهد داشت (۱۳ * ۱۲۸) یا ۵۲ کلمهٔ ۳۲ بیتی در حالیکه یک کلید ۲۵۶ بیتی به ۱۹۲۰ بیت از جنس‌کلید (۱۵ * ۱۲۸) یا ۶۰ کلمهٔ ۳۲ بیتی نیاز خواهد داشت. بنابراین، مکانیزم بسط کلید با کلید رمز ۱۲۸، ۱۹۲ و یا ۲۵۶ بیتی شروع می‌شود و یک کلید بسط‌یافتهٔ ۱۴۰۸، ۱۶۶۴ و یا ۱۹۲۰ بیتی، به ترتیب تولید می‌کند. کلید رمز اصلی اولین بخش کلید بسط یافته را اشغال می‌کند و برای تولید بقیهٔ کلید جدید مورد استفاده قرار می‌گیرد. حاصل، یک کلید بسط‌یافته است که می‌تواند به عنوان ۱۱، ۱۳ یا ۱۵ کلید جداگانه از آنها استفاده شود و هر کدام برای یک عمل AddRoundKey مورد استفاده قرار گیرد. این کلیدها، کلیدهای مرحله‌ای می‌باشند. نمودار زیر (شکل ۴-۱۷) یک مثال که از کلید رمز ۱۹۲ بیتی استفاده می‌کند را نشان می‌دهد (Nk =۶).

[1] Round Keys
[2] Expanded key
[3] Key material

Expanded Key:	W_0	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	...	W_{44}	W_{45}	W_{46}	W_{47}	W_{48}	W_{49}	W_{50}	W_{51}
Round keys:	Round key 0				Round key 1				Round key 2				...	Round key 11				Round key 12			

شکل ۴-۱۷ استفاده از کلید رمز ۱۹۲ بیتی

تبدیل AddRoundKey (به نام Round Key addition در Rijndael) فقط هرکلید مرحله‌ای را به ترتیب توسط یک عمل بیتی یای انحصاری (XOR) به حالت اعمال می‌کند. بیاد داریم که هر کلید مرحله‌ای هم‌اندازهٔ بلوک می‌باشد.

به طور خلاصه، رمزکنندهٔ AES (Rijndael) خود شامل تعدادی مراحل تشکیل‌شده از تنها چند تابع می‌باشد:

- SubBytes : مقدار یک کلمه در یک حالت را می‌گیرد و آن را با مقدار دیگری بوسیلهٔ یک S-box از قبل‌تعریف‌شده جایگزین می‌کند.
- ShiftRows : به صورت دورانی هر سطح در حالت را به تعداد بایتی از پیش‌تعریف‌شده جابه‌جا می‌کند.
- MixColumns : مقدار یک ستون چهارکلمه‌ای در یک حالت را گرفته و چهار مقدار را با استفاده از یک تابع ریاضی از پیش تعریف شده، تغییر می‌دهد.
- AddRoundKey : یک کلید را که هم طول بلوک است با استفاده از یک کلید بسط داده شدهٔ مشتق شده از کلید رمز اصلی XOR می‌کند.

به عنوان آخرین نمایش از نحوهٔ عمل AES، شکل ۴-۱۸ یک "شبه برنامه"[1] است که عمل این الگوریتم را نشان می‌دهد.

```
Cipher (byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
  byte state[4,Nb]
  state = in

  AddRoundKey(state, w)

  for round = 1 step 1 to Nr-1
    SubBytes(state)
    ShiftRows(state)
    MixColumns(state)
    AddRoundKey(state, w+round*Nb)
  end for
  SubBytes(state)
  ShiftRows(state)
  AddRoundKey(state, w+Nr*Nb)
  out = state
end
```

[1] Pseudocode

شکل ۴- ۱۸ شبه الگوریتم نمایش‌دهندهٔ الگوریتم AES

در این کد:

- in[] و out[] آرایه‌های ۱۶ بایتی به ترتیب نشان دهندهٔ plaintext و ciphertext می‌باشند. (برطبق مشخصات، هر دوی این آرایه‌ها در حقیقت $4 * N_b$ بایت طول دارند اما در AES، $N_b = 4$ است).
- state[] یک آرایهٔ دو بعدی حاوی ۴ سطر و ۴ ستون است. (بر طبق مشخصات AES، این آرایه ۴ سطر در N_b ستون می‌باشد).
- w[] یک آرایه حاوی مواد اولیهٔ کلید (key material) می‌باشد و $4 * (N_r + 1)$ کلمه طول دارد (بر طبق مشخصات، مضروب فیه دقیقاً Nb می‌باشد).
- AddRoundKey()، SubBytes()، ShiftRows() و MixColumns() توابعی هستند که نشان‌دهندهٔ تبدیلات مشخص و منحصربه‌فرد هستند.

۴-۴-۲-۳. امنیت AES

AES سه اندازهٔ کلید را مشخص می‌کند؛ کلیدهای ۱۲۸، ۱۹۲ و ۲۵۶ بیتی. این بدین معنی است که حدوداً

$۳/۴ * ۱۰^{۳۸}$ کلید ۱۲۸ بیتی ممکن،

$۶/۲ * ۱۰^{۵۷}$ کلید ۱۹۲ بیتی ممکن و

$۱/۱ * ۱۰^{۷۷}$ کلید ۲۵۶ بیتی ممکن

وجود دارند.

در مقایسه، کلیدهای DES ۵۶ بیت طول دارند که تقریباً $۷/۲ * ۱۰^{۱۶}$ کلید ممکن DES را منجر می‌شود. بنابراین حدود $۱۰^{۲۱}$ برابرکلید AES ۱۲۸ بیتی نسبت به کلیدهای ۵۶ بیتی DES وجود دارند. دراین‌صورت اگر فرض شود که کسی بتواند ماشینی بسازد که یک کلید DES را در یک ثانیه بیابد (!!) (یعنی $۲^{۵۵}$ کلید را در هر ثانیه بررسی کند)، برای چنین ماشینی تقریباً ۱۴۹ هزار بیلیون (۱۴۹ تریلیون) سال طول خواهد کشید که یک کلید ۱۲۸ بیتی AES را بیابد. برای اینکه بزرگی چنین زمانی را متصور شویم، کافیست بدانیم که گمان می‌شود عمر جهان کمتر از ۲۰ بیلیون سال باشد.

این الگوریتم همچنین در برابر سایر حمله‌های شناخته شده هم مقاومت خوبی نشان داده است؛ با این وجود هیچکس نمی‌تواند مطمئن باشد که چه مدت زمان، AES- یا هر الگوریتم رمزنگاری دیگر- امن خواهد ماند.

NIST همچنان به بررسی این الگوریتم ادامه خواهد داد و انتظار دارد که انجمن رمز از طریق کنفرانسهای گوناگون نظیر CRYPTO، EUROCRYPT، ASIACRYPT و کارگاه FSE[1] به تحلیل AES ادامه دهد.

۴-۴-۲-۴. پیاده‌سازی AES

تاکنون پیاده‌سازی‌های گوناگون سخت‌افزاری و نرم‌افزاری از این الگوریتم صورت گرفته است و نشان داده است که در هر دو زمینه، الگوریتمی کارآ می‌باشد. پیاده‌سازی‌های انجام‌شده از این الگوریتم در آمریکا، تحت قوانین صدور فن‌آوری رمزنگاری، قابل صدور هستند (برای اطلاعات دقیقتر باید با وزارت تجارت آمریکا دایرهٔ کنترل صادرات ارتباط برقرار نمود. http://www.bxa.doc.gov/Encryption/) و اگر کسی شخصاً مایل به پیاده‌سازی آن باشد؛ راهنمایی‌ها و "مقادیر آزمایش"[2] و کد نمونه در دسترس می‌باشد.

۴-۴-۳. خلاصه‌ای از رمزکننده‌های بلوکی

جدول ۴-۹ خلاصه‌ای از خصوصیات رمزکننده‌های بلوکی (قبل از پروژهٔ AES) را نشان می‌دهد (اندازهٔ بلوک و کلید به بیت داده شده‌اند).

جدول ۴-۹ خلاصه‌ای از خصوصیات رمزکننده‌های بلوکی

Name	Version	Author(s)	Block	Key	Rounds	Attack(s)
DES	(77)	IBM/NSA	64	56	16	K:43/19/13 [M94]
	3-DES (77)	Diffie, Hellman	64	168	48	K:2/112/56
	2k3-DES (78)	Tuchmann	64	112	48	K:n/120-n/n [OW91], C:56/56/56/ [M79]
FEAL-N	(87-90)	Miyaguchi, ..	64	128	N	K:2/./.(4), C:4/./.(8) [AO96]

[1] Fast Software Encryption Workshop
[2] Test values

RC2	(89)	Rivest	64	8-1024	18	C:64/64/.(16) [KRRR98]
Khufu	(90)	Merkle	64	512	8s, s>1	C:52/./.(26) [BBS99]
Khafre	(90)	Merkle	64	64t, t>0	8s, s>1	C:52/./.(26) [BBS99]
IDEA	(91)	Lai, Massey, Murphy	64	128	8,5	C:64/112/32(4,5) [BBS99]
LOKI	(90)	Brown, Pieprzyk, Seberry	64	64	16	C:54/./.(14), K:62/./.(11)
	(91)	Brown, Kwan, Pieprzyk, Seberry	64	64	16	C:58/./.(13) [K94] , K:60/./.(11) [SF97]
SAFER	K (93)	Massey	64	64,128	6,10	C:45/./32(5) [KB96]
	SK (95)	Massey, Knudsen	64	40,64,128	8,10	?
Blowfish	(93)	Schneier	64	32-448	16	?
RC5	32/12/k (94)	Rivest	64	8s, s<256	12	C:54/./. [KM96]
	64/16/16 (94)	Rivest	128	8s, s<256	16	C:83/./. [KM96] , C:123/./.(24) [KM96]

CAST-128	(95)	Adams	64	40-128	12, 16	?
SHARK	(96)	Rijmen, Daemen, Preneel, Bosselaers, de Win	64	128	6	?
SQUARE	(97)	Daemen, Knudsen, Rijmen	128	128	8	CP:32/72/32(6) [DKR97]
MISTY	1 (97)	Matsui	64	128	8	?
	2 (97)	Matsui	64	128	12	?
ICE	(97)	Kwan	64	64	16	CP:62/62/30 [VRKR98]
Skipjack	(98)	NSA?	64	80	32	[BS98]
Rainbow	(98)	Lee, Kim	128	128	7	?
Name	**Version**	**Author(s)**	**Block**	**Key**	**Rounds**	**Attack(s)**

نشان‌گذاری جدول:

Name نام رمزکنندهٔ بلوکی

Version نام (سال) نسخه

Author(s) نام طراح (ها)

Block طول بلوک به بیت

Key طول کلید به بیت

Rounds تعداد مراحل (rounds) رمزکننده

Attack(s)

K:a/b/c بیان می‌کند که بهترین حملهٔ known plaintext به۲a plaintext/ciphertext ، مقدار کار۲b رمزکردن و۲c کلمهٔ حافظه نیاز دارد.

C:a/b/c بیان می‌کند که بهترین حملهٔ chosen plaintext به۲a plaintext/ciphertext ، مقدار کار۲b رمزکردن و۲c کلمهٔ حافظه نیاز دارد.

یک "." بدین معنی است که مقدار نیاز به منبع قابل صرفنظر است یا نامعلوم می‌باشد.

(r) : تعداد مراحل حمله. در صورت خالی‌بودن، r=Rounds.

[SA] : مقاله‌ای که حمله در آن شرح داده شده‌است.

? : حمله‌ای شناخته نشده‌است.

نکته! No known attacks (?) لزوماً به معنای امن‌بودن رمزکنندهٔ بلوکی نمی‌باشد.

مراجع برای جدول قبلی:

[A97]

C. Adams, "Constructing Symmetric Ciphers using the CAST Design Procedure", *Designs, Codes and Cryptography, vol.12, no.3, November 1997, pp.283-316 (see also Selected Areas in Cryptography, Kluwer Academic Publishers, 1997, pp.71-104).*

[A97-2]

C. Adams, "The CAST-128 Encryption Algorithm", *RFC 2144, May 1997.*

[AO96]

K. Aoki and K. Ohta, ``Differential-linear cryptanalysis of FEAL-8," *IEICE Transactions on fundamentals of electronics, communications, and computer sciences*, Vol. E79-A, No. 1, January 1996.

[BS93]

E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

[BS98]

E. Biham, A. Biruykov, A. Shamir *``Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials"*

[BBS99]

E. Biham, A. Biruykov, A. Shamir ``Miss in the middle attacks on IDEA, Khufu and Khafre," *Fast Software Encryption '99, LNCS.*

[DKR97]

J. Daemen, L.R. Knudsen, and V. Rijmen, ``The block cipher SQUARE," *Fast Software Encryption, LNCS 1267,* E. Biham, Ed., Springer-Verlag, pp. 149-165.

[GC94]

H. Gilbert and P. Chauvaud, ``A chosen-plaintext attack of the 16-round Khufu cryptosystem," *Advances in Cryptology, Proceedings Crypto'94, LNCS 839,* Y.~Desmedt, Ed., Springer-Verlag, 1994, pp. 359-368.

[KL98]

C-H.~Lee, J-S.~Kim, ``Rainbow". Contains postscript paper and source code.

[K94]

L.R. Knudsen, ``Block ciphers - analysis, design and applications," *Ph.D. Thesis, DAIMI PB 485,* Aarhus University, 1994.

[KB96]

L.R.~Knudsen and T.A.~Berson, ``Truncated differentials of SAFER," *Fast Software Encryption, LNCS 1039,* D. Gollmann, Ed., Springer-Verlag, 1996, pp. 15-26.

[KM96]

L.R.~Knudsen and W.~Meier, ``Improved differential attack on RC5," *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109,* N. Koblitz, Ed., Springer-Verlag, 1996, pp. 216-228.

[KRRR98]

L.R. Knudsen, V. Rijmen, R.L. Rivest and M.J.B. Robshaw, ``On the design and security of RC2," *Fast Software Encryption, LNCS 1372,* S. Vaudenay, Ed., Springer-Verlag, 1998, pp. 206-221.

[M94]

M. Matsui, ``Linear cryptanalysis method for DES cipher," *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765,* T. Helleseth, Ed., Springer-Verlag, 1994, pp. 386-397.

[M79]

R.C. Merkle, *Secrecy, authentication, and public-key systems,* UMI Research Press, Ann Arbor, Michigan, 1979.

[SF97]

K. Sakurai and S. Furuya, ``Improving linear cryptanalysis of LOKI91 by probabilistic counting method," *Fast Software Encryption, LNCS 1267,* E. Biham, Ed., Springer-Verlag, pp 114-133.

[OW91]

P.C. van Oorschot and M. Wiener, ``A known-plaintext attack on two-key triple encryption,'' *Advances in Cryptology, Proceedings Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer-Verlag, 1991, pp. 318-325.

[VRKR]

B. Van Rompay, L.R. Knudsen and V. Rijmen, ``Differential cryptanalysis of the ICE encryption algorithm,'' *Fast Software Encryption, LNCS 1372*, S. Vaudenay, Ed., Springer-Verlag, 1998, pp. 270-283.

به دلیل اهمیت پروژهٔ AES ، الگوریتم‌های پیشنهاد شده برای این پروژه به صورت جداگانه در جدول ۴-۱۰ لیست شده‌اند (همهٔ این الگوریتم‌ها می‌بایست اندازهٔ بلوک ۱۲۸ بیتی و اندازهٔ کلید ۱۲۸، ۱۹۲ و ۲۵۶ بیتی را پشتیبانی می کردند).

جدول ۴-۱۰ ۱۵ الگوریتم پیشنهادی برای AES

Name	Version	Author(s)	Block	Key	Rounds	Attack(s)
CAST-256	1999	Adams	128	128,160,192, 224,256	48	?
CRYPTON	1999	Lim	128	64-256	12	C:32/56/32 (6) [DB99]
DEAL	1998	Knudsen, Outerbridge	128	128,192,256	6,8	, [Luc98] [KS99]
DFC	1998	Vaudenay et al	128	128,192,256	8	[KR99]
E2	1998	Aoki, Kanda, Matsumoto, Moriai, Ohta, Ookubo, Takashima, Ueda	128	128,192,256	12	C:100/./. (8) [MT99]
FROG	1998	Georgoudis, Leroux, Chaves	64-1024	40-1000	8	[W98]

Hasty Pudding	1999	R. Schroeppel	any number of bit	0-65536	8	?
LOKI97	1998	Brown, Pieprzyk	128	128,192,256	16	K:56/./., C:56/./., [RK98]
MARS	1999	IBM	128	128-448	32	[Saar98]
Magenta	1998	Deutsche Telekomm	128	128	6,8	[BBFKS]
RC6	1998	Rivest, Robshaw, Sidney, Yin	128	64-1024	20	?
RIJNDAEL	1999	Daemen, Rijmen	128,192,256	128,192,256 (expandable to multiple of 32 bit)	10,12,14	?
SAFER+	1998	Massey, Khachatrian, Kuregian	128	128,192,256	8,12,16	[KSW99]
SERPENT	1998	Anderson, Biham, Knudsen	128	128-256	32	?
TWOFISH	1998	Schneier, Kelsey, Whiting, Wagner, Hall, Ferguson	128	up to 256	16	MM99

نشان‌گذاری این جدول همانند نشان‌گذاری جدول قبلی می‌باشد.

مراجع برای جدول :

[NES2000]

B. Preneel et al.: Comments by the NESSIE Project on the AES Finalists

[A98]

C. Adams: The CAST-256 Encryption Algorithm

[GM2000]

H. Gilbert, M. Minier: A collision attack on 7 rounds of Rijndael

[KS2000]

J. Kelsey, B. Schneier: MARS Attacks! Preliminary Cryptanalysis of Reduced-Round MARS Variants

[KKS2000]

T. Kohno, J. Kelsey, B. Schneier: Preliminary Cryptanalysis of Reduced-Round Serpent

[BK2000]

E. Biham, N. Keller: Cryptanalysis of Reduced Variants of Rijndael

[Fe2000]

N. Ferguson, J. Kelsey, B. Schneier, M. Stay, D. Wagner, D. Whiting: *"Improved Cryptanalysis of Rijndael"*, FSE2000

[Lu2000]

S. Lucks: Attacking Seven Rounds of Rijndael under 192-bit and 256-bit Keys

[ABK98]

R. Anderson, E. Biham, L. Knudsen: SERPENT

[BBFKS]

E. Biham, A. Biryukov, N. Ferguson, L. Knudsen, B. Schneier, A. Shamir: Cryptanalysis of MAGENTA (pdf)

[BP98]

L. Brown, J. Pieprzyk: Introducing the new LOKI97 Block Cipher

[IBM98]

Burwick, Coppersmith, D'Avignon, Gennaro, Halevi, Jutla, Matyas Jr., O'Connor, Peyravian, Safford, Zunic: MARS - a candidate cipher for AES

[Gil2000]

H. Gilbert, H. Handschuh, A. Joux, S. Vaudenay: "A Statistical Attack on RC6", FSE2000

[CL98]

Cylink Corporation: SAFER+ (No link, LK, 11.08.99.)

[DR98]

J. Daemen, V. Rijmen: AES Proposal: Rijndael

[DR00]

J. Daemen, V. Rijmen: Answer to "new observations on Rijndael"

[SK98]

B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson: On the Twofish Key Schedule

[WK99]

D. Whiting, J. Kelsey, B. Schneier, D. Wagner, N. Ferguson, and C. Hall: Further Observations on the Key Schedule of Twofish

[DB99]

C. D'Halluin, G. Bijnens, V. Rijmen, B. Preneel: *"Attack on 6 rounds of Crypton"*, FSE'99, LNCS.

[GLC]

D. Georgiadis, D. Leroux, B.S. Chaves: The FROG Encryption Algorithm

[KSW99]

J. Kelsey, B. Schneier, D. Wagner: Key schedule weaknesses in SAFER+

[KS99]

J. Kelsey, B. Schneier: Keyschedule Cryptanalysis of DEAL, SAC 99

[DEAL]

L. Knudsen: DEAL: A 128-bit Block Cipher

[KM99]

L. Knudsen, W. Meier: Correlations in RC6

[KR99AL]

L. Knudsen, V. Rijmen: *"On the Decorrelated Fast Cipher (DFC) and its theory"*, FSE'99, LNCS.

[LK00]

L. Knudsen: Trawling Twofish (revisited)

[Luc98]

S. Lucks: On the Security of the 128-bit Block Cipher DEAL

[MT99]

M. Matsui, T. Tokita: *"Cryptanalysis of a reduced version of the block cipher E2"*, FSE'99, LNCS.

[E2]

Nippon Telegraph and Telephone Corporation: The 128-Bit Block Cipher E2

[Lim98]

C. H. Lim: CRYPTON

[MM99]

F. Mirza, S. Murphy: An Observation on the Key Schedule of Twofish

[SM00]

S. Murphy: The Key Separation of Twofish

[MR00]

S. Murphy, M. Robshaw: New Observations on Rijndael

[RK98]

V. Rijmen, L.R. Knudsen: Weaknesses in LOKI97
(pdf)(pdf)

[RRSY]

R. Rivest, M.J.B. Robshaw, R. Sidney, Y.L. Yin: The RC6 Block Cipher
(pdf) . See also here.

[Saar98]

M-J. Saarinen: Equivalent keys in MARS
M-J. Saarinen: A note regarding the hash function use of MARS and RC6

[TF98]

Schneier, Kelsey, Whiting, Wagner, Hall, Ferguson: Twofish: A 128-bit Block Cipher

[S98]

R. Schroeppel: The Hasty Pudding Cipher

[BF2000]

E. Biham, V. Furman: Impossible Differential on 8-Round MARS' Core

[V98]

S. Vaudenay et al.: DFC

[W98]

D. Wagner, N. Ferguson, and B. Schneier: Cryptanalysis of Frog

سرانجام، جدول ۴-۱۱ پنج الگوریتم فینالیست پروژهٔ AES را نشان می‌دهد.

جدول ۴-۱۱ پنج الگوریتم فینالیست پروژهٔ AES

Name	Author(s)	Report(s)
MARS	IBM (11 authors)	BF2000, KS2000, Sub.stat."Tweak"
RC6	Rivest, Robshaw, Sidney, Yin	, Gil2000, Sub.stat.KM99
RIJNDAEL	Daemen, Rijmen	, BK2000, Lu2000, MR00, DR00, GM2000 Sub.stat.

SERPENT	Anderson, Biham, Knudsen	, Sub.stat.KKS2000
TWOFISH	Schneier, Kelsey, Whiting, Wagner, Hall, Ferguson	, SM00, LK00, WK99, SK98 MM99 Sub.stat.

"Sub.stat." اظهارات نهایی ارائه‌دهندگان می‌باشد.

۴-۵. رمزکننده‌های جریانی

"رمزکننده‌های جریانی"[1] یک دستهٔ مهم از الگوریتم‌های رمزنگاری متقارن می‌باشند که روی یک بیت، بایت و یا کلمهٔ (کامپیوتر) از پیام plaintext در یک زمان عمل می‌کنند و به این دلیل در کاربردهایی که با یک جریان داده روبرو هستند مناسب می‌باشند. در واقع یک رمزکنندهٔ جریانی شامل یک ماشین حالت است که در هر گذر حالت یک بیت از اطلاعات را خارج می‌کند. این جریان بیت‌های خروجی معمولاً "کلید جاری"[2] یا "جریان‌کلید"[3] نامیده می‌شود. عمل رمزکردن می‌تواند تنها با XOR کردن جریان‌کلید و پیام plaintext پیاده‌سازی شود.

رمزکننده‌های جریانی به دلیل داشتن ساختار ساده‌تر می‌توانند طوری طراحی شوند که از هر رمزکنندهٔ بلوکی سریعتر باشند. همانطور که قبلاً نیز اشاره شده‌است، این رمزکننده‌ها بر خلاف رمزکننده‌های بلوکی، که یک plaintext مشخص به یک ciphertext معین رمز می‌شود، بر حسب این که در چه زمانی در خلال فرآیند رمز کردن با یک plaintext برخورد شود ciphertextهای متفاوتی تولید خواهد شد.

در این رمزکننده‌ها تعدادی از مکانیزم‌های "پس‌خورد" (فیدبک) پیاده‌سازی می‌شود، به نحوی که کلید به طور دائمی تغییر می‌کند. ماشین حالت به‌کاررفته در این رمزکننده‌ها بیش از یک مولد اعداد شبه‌تصادفی نیست. در این صورت، رمزکردن نیز چیزی جز ترکیب این اعداد با plaintext (مثلاً XOR کردن) نمی‌باشد. به عنوان مثال، یک روش تولید اعداد شبه‌تصادفی می‌تواند رمزکردن مکرر خروجی یک رمزکنندهٔ بلوکی باشد. نوعاً برای این رمزکننده‌ها ساختارهای خیلی پیچیده (دارای جزئیات زیاد) طراحی می‌شوند تا درعوض در هنگام پیاده‌سازی به سرعت بالاتری دست پیدا کنند. هر چه اعداد تصادفی بهتر و نزدیک به واقعیت تولید شوند، رمزکنندهٔ جریانی استفاده‌کننده از آن نیز امن‌تر خواهد بود.

رمزکننده‌های جریانی عموماً در سخت‌افزار از رمزکننده‌های بلوکی سریع‌تر هستند و مدار سخت‌افزاری با پیچیدگی کمتری دارند. این رمزکننده‌ها در مواردی که میانگیری[4] محدود است یا زمانی که کاراکترها باید در هنگام دریافت به صورت جداگانه مورد پردازش قرار گیرند؛ خیلی مناسب و در بعضی حالات (مثلاً در بعضی کاربردهای مخابراتی[5]) الزامی هستند. به دلیل این که رمزکننده‌های جریانی انتشار خطا ندارند و یا دارای انتشار خطای محدود هستند، می‌توانند در مواقعی که خطاهای ارسال خیلی محتملند، سودمند باشند.

بر خلاف سایر دسته‌های الگوریتم‌های رمزنگاری که غالباً الگوریتم‌های استاندارد شده دارند، در این دسته، چنین چیزی به چشم نمی‌خورد (مثلاً در گروه رمزکننده‌های بلوکی، AES (Rijndael) به عنوان استاندارد رمزنگاری دولت آمریکا انتخاب شده است). البته جدیداً اتحادیهٔ اروپا در برنامهٔ IST[6] پروژه‌ای را تحت عنوان "روش‌های جدید اروپایی برای امضاء،

[1] Stream Ciphers
[2] Running key
[3] Keystream
[4] Buffering
[5] Telecommunications
[6] Information Society Technologies

جامعیت و رمزکردن" (NESSIE[1]) با مشارکت کشورهای ایتالیا، آلمان، بلژیک، انگلیس، فرانسه، نروژ و اسرائیل در دست انجام دارند که در پی ارائهٔ استانداردهایی برای دسته‌های مختلف الگوریتم‌های رمزنگاری می‌باشد. از اهداف این پروژه همکاری در فاز نهایی فرآیند استانداردسازی AES (سازماندهی شده توسط NIST در ایالات‌متحده) بوده است؛ با این حال، خود مستقلاً یک فراخوان باز برای مجموعهٔ گسترده‌ای از Primitive ها برای مهیاکردن محرمانگی، جامعیت داده و اعتبارسنجی انجام داد. این Primitive ها شامل رمزکننده‌های بلوکی، رمزکننده‌های جریانی و توابع درهم‌سازی، الگوریتم‌های MAC[2]، روش‌های امضای دیجیتالی و بالاخره روش‌های رمزکردن کلید عمومی می‌باشد. هدف پروژه این است که نتایج آن به صورت گسترده منتشر شود و یک توافق عام بر پایهٔ این نتایج، با استفاده از زمینه‌های مناسب موجود صورت گیرد.

مقصود نهایی پروژهٔ NESSIE ابقاء جایگاه قوی تحقیق اروپا به همراه تقویت جایگاه صنعت رمزنگاری اروپا می‌باشد. زمان انجام پروژه از ۲۰۰۰/۱/۱ تا ۲۰۰۲/۱۲/۳۱ یعنی به مدت ۳۶ ماه می‌باشد که در حال حاضر در دست اجرا است. این پروژه در واقع الگوریتم‌های رمزنگاری را که برای پشتیبانی تجارت الکترونیکی، e-government و امضاهای الکترونیکی حیاتی هستند، ارزیابی می‌کند. فازهای اولیهٔ پروژه انجام شده‌اند که در آنها از ۱۰ کشور مختلف در سراسر جهان، ۴۲ الگوریتم رمز ارائه شد و بر اساس ارزیابی‌های انجام شده، این تعداد به ۲۴ الگوریتم کاهش یافت. پس از پایان پروژه یک مجموعهٔ نهایی از توصیه‌نامه‌ها با بهترین الگوریتم‌های رمزنگاری در هر بخش منتشر خواهند شد. همچنین قصد بر این است که این الگوریتم‌ها به بدنه‌های استانداردسازی نظیر ISO، IETF و IEEE وارد شوند.

برای مرحلهٔ دوم در بخش رمزکننده‌های جریانی، رمزکننده‌های زیر انتخاب شده‌اند:

- SOBER-t16 و SOBER-t32 از Qualcomm International استرالیا
- SNOW از Lund Univ. سوئد
- BMGL از Royal Institute of Technology (Ericsson Research, Stockholm) سوئد

ارزیابی‌ها یک فرآیند کاملاً باز بر پایهٔ معیارهای ارزیابی منتشرشده می‌باشند. همهٔ نامزدها در فاز دوم در یک کارگاه در نوامبر ۲۰۰۲ مطرح خواهند شد و انتخاب نهایی در دسامبر ۲۰۰۲ انجام خواهد شد.

۴-۵-۱. تفاوت عمدهٔ رمزکننده‌های جریانی با رمزکننده‌های بلوکی

رمزکننده‌های بلوکی، plaintext را در بلوک‌های نسبتاً بزرگ (مثلاً n بزرگتر از ۶۴ بیت) پردازش می‌کنند و **یک** تابع برای رمزکردن بلوک‌های متوالی استفاده می‌شود؛ بنابراین رمزکننده‌های بلوکی بدون حافظه هستند. در مقابل،

[1] New European Schemes for Signature, Integrity and Encryption
[2] Message Authentication Code

رمزکننده‌های جریانی plaintext را در بلوک‌هایی حتی به کوچکی یک بیت پردازش می‌کنند و تابع رمزکننده در زمانی که plaintext پردازش می‌شود، می‌تواند تغییر کند؛ بنابراین گفته می‌شود که رمزکننده‌های جریانی حافظه‌دار هستند. گاهی اوقات به آنها، "رمزکننده‌های‌حالت"[1] نیز گفته می‌شود چراکه رمزکردن، نه تنها به کلید و plaintext، بلکه به حالت فعلی نیز بستگی دارد. این تمایز بین رمزکننده‌های بلوکی و جریانی کاملاً مشخص‌کننده نیست؛ چرا که اضافه‌کردن مقداری حافظه به یک رمزکنندهٔ بلوکی (همانند استفاده از آنها در حالت CBC[2]) باعث ایجاد یک رمزکنندهٔ جریانی با بلوک‌های بزرگ می‌شود.

۴-۵-۲. دسته‌بندی رمزکننده‌های جریانی

در ادامه، به دسته‌بندی این گروه از رمزکننده‌ها اشاره می‌شود.

۴-۵-۲-۱. one-time pad

یک رمزکنندهٔ Vernam روی ارقام دودویی چنین تعریف می‌شود:

$c_i = m_i \oplus k_i$, for i = 1, 2, 3, … .

که m_1، m_2، m_3 و ... ارقام plaintext، k_1، k_2، k_3 و ... (Keystream) ارقام کلید و c_1، c_2، c_3 و ... ارقام ciphertext می‌باشند و $\oplus$ تابع XOR (جمع دودویی در پایهٔ ۲) می‌باشد. رمزگشایی هم بوسیلهٔ $m_i = c_i \oplus k_i$ تعریف می‌شود. اگر ارقام Keystream به صورت مستقل و تصادفی تولید شوند، رمزکنندهٔ Vernam یک one-time pad نامیده می‌شود و به صورت **غیرمشروط** امن است. از آنجایی‌که Keystream هم‌طول پیغام plaintext می‌باشد و بیت به بیت با plaintext، XOR می‌شود تا ciphertext تولید شود، یک دشمن (متخاصم) با منابع پردازشی بی‌نهایت، در صورت دیدن ciphertext تنها می‌تواند plaintext را حدس بزند. چنین رمزکننده‌ای محرمانگی کاملی را ارائه می‌دهد و تحلیل one-time pad به عنوان یکی از پایه‌های رمزنگاری مدرن محسوب می‌شود.

یک اشکال (مانع) در رمزکنندهٔ one-time pad این است که کلید باید هم طول plaintext باشد که این مطالب مشکل توزیع و مدیریت کلید را افزایش می‌دهد. این مشکل، طراحی رمزکننده‌های جریانی را وادار می‌کند که Keystream به صورت شبه تصادفی از یک کلید محرمانهٔ کوچکتر تولید شود، با این نیت که Keystream نزد یک متخاصم با توان پردازشی محدود، تصادفی به نظر برسد. چنین رمزکننده‌های جریانی امنیت غیرمشروطی را ارائه نمی‌دهند اما امید است که آنها در برابر روش‌های محاسباتی امن باشند. به صورت معمول رمزکننده‌های جریانی به دو دستهٔ همگام[3] و "خودهمگام‌شونده"[4] یا ناهمگام تقسیم می‌شوند.

[1] State Ciphers
[2] Cipher Block Chaining
[3] Synchronous
[4] Self-synchronizing

۴-۵-۲-۲. رمزکننده‌های جریانی همگام

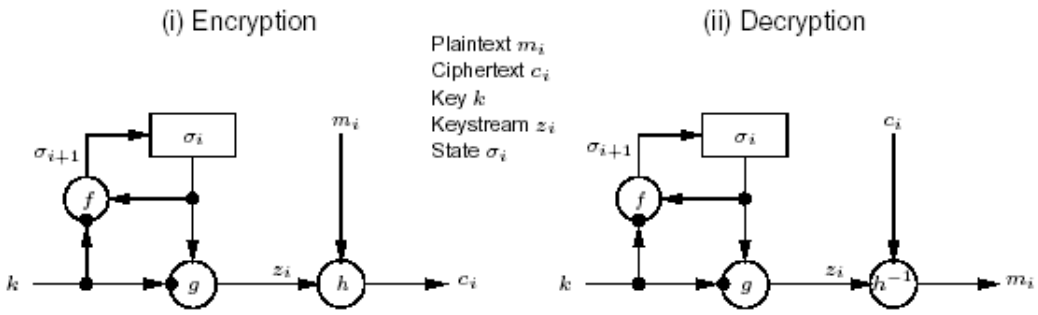
یک رمزکنندهٔ جریانی همگام رمزکننده‌ای است که Keystream به صورت مستقل از پیام plaintext و ciphertext تولید می‌شود. فرآیند رمزکردن یک رمزکنندهٔ جریانی می‌تواند توسط معادلات زیر تشریح شود:

$\sigma_{i+1} = f(\sigma_i, k)$,

$z_i = g(\sigma_i, k)$,

$c_i = h(z_i, m_i)$,

که σ_0 حالت اولیه است و می‌تواند از کلید k تعیین شود و f "تابع حالت بعدی"[1]، g تابعی که z_i Keystream را تولید می‌کند و h هم تابع خروجی است که Keystream را با m_i plaintext برای تولید c_i ciphertext ترکیب می‌کند. فرآیندهای رمزکردن و رمزگشایی در شکل ۴-۱۹ نشان داده شده‌اند.



شکل ۴-۱۹ مدل عمومی یک رمزکنندهٔ جریانی همگام

حالت OFB[2] یک رمزکنندهٔ بلوکی مثالی از رمزکنندهٔ جریانی همگام می‌باشد.

خواص رمزکننده‌های جریانی همگام عبارتند از:

- *نیازهای همزمانی*

در چنین رمزکننده‌ای فرستنده و گیرنده باید همگام باشند، یعنی استفاده از یک کلید و عمل در یک مکان (حالت) داخل آن کلید به‌منظور رمزکردن مناسب صورت پذیرد. اگر همزمانی در نتیجهٔ درج یا حذف ارقام ciphertext در خلال ارسال از بین برود، رمزگشایی خراب می‌شود و تنها از طریق تکنیک‌های اضافی برای دوباره-همزمانی می‌تواند به حال اول برگردد.

[1] next-state function
[2] Output FeedBack Mode

- *عدم انتشار خطا*

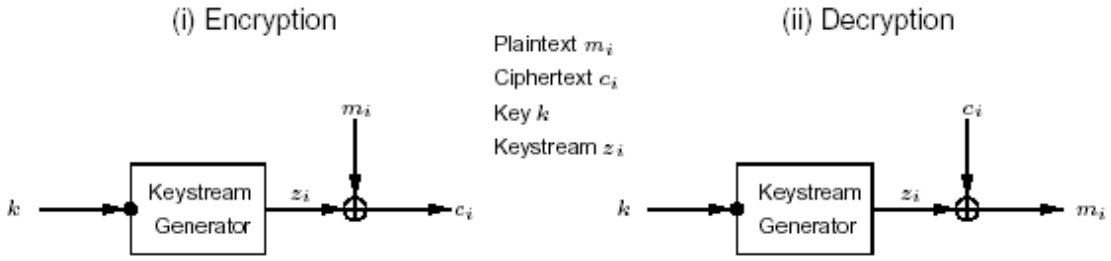
اگر یک رقم ciphertext در خلال ارسال تغییرکند (اما حذف نشود) بر رمزگشایی سایر ارقام ciphertext تأثیری نمی‌گذارد.

- *حمله‌های فعال*[1]

در نتیجهٔ خاصیت اول، حذف، درج یا تکرار ارقام ciphertext بوسیلهٔ یک متخاصم فعال، باعث از بین رفتن بلافاصلهٔ همزمانی می‌شود و بنابراین احتمالاً می‌تواند توسط رمزگشا تشخیص داده شود. به عنوان نتیجهٔ خاصیت دوم یک دشمن فعال احتمالاً می‌تواند به ایجاد تغییر در ارقام انتخاب شدهٔ ciphertext قادر باشد و دقیقاً بداندکه این تغییرات چه تأثیری بر plaintext دارد. این نشان می‌دهد که مکانیزم‌های اضافی برای "اعتبارسنجی منبع داده"[2] و ضمانت جامعیت داده باید به‌کار گرفته شوند.

- *رمزکننده‌های جریانی افزایشی*

اغلب رمزکننده‌های جریانی که تا کنون ارائه شده‌اند، "رمزکننده‌های جریانی افزایشی"[3] هستند. یک رمزکنندهٔ جریانی افزایشی یک رمزکنندهٔ جریانی همگام است که ارقام Keystream، plaintext و ciphertext دودویی هستند و تابع خروجی h تابع XOR می‌باشد. رمزکننده‌های جریانی افزایشی دودویی در شکل ۴-۲۰ نشان داده شده‌اند. با توجه به این شکل مولد Keystream تشکیل شده است از تابع حالت‌بعدی f و تابع g (شکل۴-۱۹ را ببینید) که همچنین به نام "مولد کلید جاری"[4] شناخته می‌شود.



شکل ۴-۲۰ مدل عمومی یک رمزکنندهٔ جریانی افزایشی دودویی

۴-۵-۲-۳. رمزکننده‌های جریانی "خود همگام شونده"

یک رمزکنندهٔ جریانی "خود همگام شونده" یا ناهمگام[5] رمزکننده‌ای است که Keystream، تابعی از کلید و تعداد ثابتی از ارقام قبلی ciphertext می‌باشد.

تابع رمزکنندهٔ یک رمزکنندهٔ جریانی خودهمگام شونده می‌تواند به وسیلهٔ معادلات زیر توصیف شود:

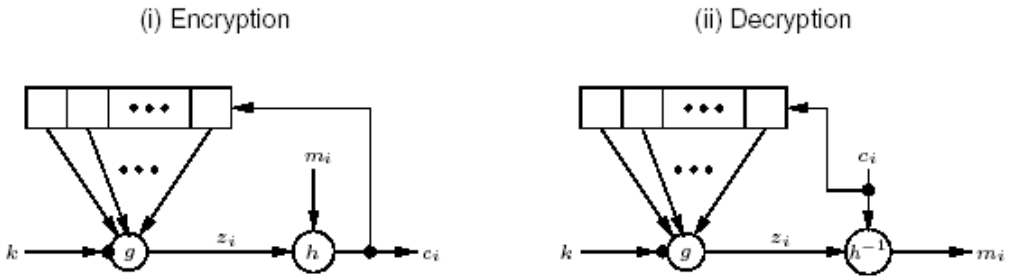
[1] Active attacks
[2] Data Origin Authentication
[3] Additive Stream Ciphers
[4] Running key generator
[5] Asynchronous

$\sigma_i = (c_{i-t}, c_{i-t+1}, \ldots, c_{i-1})$,

$z_i = g(\sigma_i, k)$,

$c_i = h(z_i, m_i)$,

که $\sigma_0 = (c_{-t}, c_{-t+1}, \ldots, c_{-1})$ حالت اولیه، k کلید، g تابعی که z_i Keystream را تولید می‌کند و h تابع خروجی است که Keystream و m_i plaintext را برای تولید c_i ciphertext ترکیب می‌کند. فرآیندهای رمزکردن و رمزگشایی در شکل ۴-۲۱ نمایش داده شده‌اند. رمزکننده‌های جریانی پر استفادهٔ خودهمگام شونده بر پایهٔ رمزکننده‌های بلوکی بنا شده‌اند که در مود پسخورد یک بیتی رمزکننده مورد استفاده واقع شده‌اند.



شکل ۴-۲۱ مدل عمومی یک رمزکنندهٔ جریانی خودهمگام‌شونده

خواص رمزکننده‌های جریانی خودهمگام شونده عبارتند از:

- *خود همگامی*[1]

خود همگامی در صورتی که ارقام ciphertext حذف شوند یا رقمی در آنها درج شود، امکان‌پذیر است، زیرا نگاشت رمزگشایی تنها به تعداد ثابتی از کاراکترهای قبلی ciphertext بستگی دارد. چنین رمزکننده‌هایی می‌توانند به صورت خودکار بعد از ازبین رفتن همگامی و با تنها تعداد ثابتی کاراکتر غیر قابل بازیافت plaintext، رمزکردن مناسب را دوباره از سر گیرند.

- *انتشار محدود خطا*

فرض کنید که حالت یک رمزکنندهٔ جریانی خودهمگام شونده به t رقم قبلی از ciphertext بستگی داشته باشد، اگر در خلال ارسال، یک رقم از ciphertext تغییر کند (یا حتی حذف و درج شود)، رمزگشایی حداکثر t رقم بعدی ciphertext ممکن است ناصحیح باشد (بعد از اینکه رمزگشایی از سرگرفته می‌شود).

- *حمله‌های فعال*

خاصیت دوم بیان می‌کند که هر تغییر در ارقام ciphertext بوسیلهٔ یک دشمن فعال باعث می‌شود که چندین رقم دیگر ciphertext به صورت ناصحیح رمزگشایی شود، بدین وسیله (در مقایسه با رمزکننده‌های همگام) شانس تشخیص داده

[1] Self-synchronization

شدن به وسیلهٔ رمزگشا بهبود پیدا می‌کند. به عنوان یک نتیجه از خاصیت اول تشخیص درج، حذف و یا تکرار ارقام ciphertext به وسیلهٔ یک دشمن فعال (نسبت به رمزکننده‌های جریانی همگام) خیلی مشکل‌تر است. این مطلب نشان می‌دهد که مکانیزم‌های اضافی باید به‌کار گرفته شوند تا اعتبارسنجی منبع داده و ضمانت جامعیت داده تأمین شود.

- *انتشار (پخش) آماری plaintext*

از آنجایی‌که هر رقم plaintext در تمامی ارقام بعدی ciphertext تأثیر دارد، خواص آماری plaintext در طول ciphertext پراکنده می‌شود. بنابراین، رمزکننده‌های جریانی خودهمگام شونده نسبت به رمزکننده‌های جریانی همگام در برابر حملات بر پایهٔ افزونگی[1] plaintext می‌توانند خیلی مقاوم‌تر باشند.

۴-۵-۳. رمزکننده‌های جریانی مهم

گذشته از پروژهٔ NESSIE الگوریتم‌های رمزکنندهٔ جریانی که تا کنون بیشتر مورد استفاده بوده‌اند، عبارتند از:

- RC4
- SEAL
- A5

الگوریتم‌های جدیدی که به جهت امن‌تر کردن الگوریتم‌های گذشته و یا اصلاحات دیگر ارائه شده‌اند عبارتند از:

- SNOW
- Scream
- SOBER-t16 ،SOBER-t32

ساختارهایی که عموماً در طراحی رمزکننده‌های جریانی مورد استفاده قرار می‌گیرند، LFSR[2]ها، FCSR[3]ها، مولدهای Shrinking و Self-Shrinking هستند که بسته به نوع کاربرد از آنها استفاده می‌شود.

اگرچه که رمزکننده‌های جریانی بر پایهٔ LFSR برای پیاده‌سازی سخت‌افزاری خیلی مناسبند، اما برای پیاده‌سازی نرم‌افزاری مناسب نیستند. این موضوع باعث شد که چندین پیشنهاد برای طراحی رمزکننده‌های جریانی که پیاده‌سازی نرم‌افزاری آنها سریع است، ارائه شود. دو تا از این رمزکننده‌ها SEAL و RC4 می‌باشند.

۴-۵-۳-۱. SEAL

الگوریتم SEAL[4] یک رمزکنندهٔ جریانی افزایشی دودویی است که در سال ۱۹۹۳ به وسیلهٔ Don Coppersmith در شرکت IBM طراحی شده است. SEAL یک تابع شبه تصادفی "بزرگ شونده در طول"[1] است که یک عدد ۳۲ بیتی n را

[1] Redundancy
[2] Linear Feedback Shift Register
[3] Feedback with Carry Shift Register
[4] Software-optimized Encryption Algorithm

به یک L Keystream بیتی تحت کنترل یک کلید محرمانهٔ ۱۶۰ بیتی a نگاشت می‌کند. در مرحلهٔ پیش‌پردازش، با استفاده از تابع مولد جدول G_a کلید در جداول بزرگتری بسط داده می‌شود که این تابع بر پایهٔ SHA-1[2] بنا شده است. پس از این پیش‌پردازش و تولید Keystream به حدود ۵ دستور ماشین به ازای هر بایت نیاز دارد و سریع‌تر از DES اجرا می‌شود.

۴-۵-۳-۲. RC4

الگوریتم RC4 یک رمزکنندهٔ جریانی است که توسط Ron Rivest در شرکت RSA Data Security طراحی شد. تا مدت‌ها به عنوان یک Trade Secret یعنی رازی که مرتبط با تولید محصولات تجاری است محسوب می‌شد تا اینکه شخصی کدی را برای یک الگوریتم در Usenet پست کرد و ادعا کرد که معادل RC4 می‌باشد. مدرک خیلی قوی موجود است که الگوریتم پست‌شده حقیقتاً مشابه RC4 می‌باشد. RC4 الگوریتم خیلی سریعی می‌باشد و امنیت آن به طور قطع مشخص نیست اما به نظر نمی‌رسد که شکستن آن ساده و پیش پا افتاده باشد. این الگوریتم کلیدهای به طول دلخواه (متغیر) می‌پذیرد. RC4 اساساً یک مولد عدد شبه تصادفی می‌باشد و خروجی مولد با جریان داده یای انحصاری (XOR) می‌شود. به این دلیل خیلی مهم است که یک کلید RC4 هرگز برای رمزکردن دو جریان دادهٔ متفاوت استفاده نشود.

دولت ایالات متحده معمولاً می‌پسندد که RC4 با طول کلید کمتر از ۴۰ بیت برای صادرات استفاده شود. کلیدهای به این کوچکی می‌توانند به راحتی توسط دولت‌ها، جنایتکاران و حتی آماتورها شکسته شوند. جالب توجه است که بدانیم نسخهٔ صادراتی SSL[3] که از RC4-40 استفاده می‌کند به وسیلهٔ حداقل دو گروه مستقل شکسته شده است و شکستن آن هم حدود ۸ روز طول کشید.

علیرغم نقاط ضعفی که RC4 دارد در شرایطی که نکات ایمنی زیر رعایت شود هنوز می‌تواند امن باشد:

- برای ایجاد "کلیدهای جلسه"[4] از کلیدهای محرمانه و IV[5] ها، از توابع درهم‌سازی استفاده شود؛
- اولین ۲۵۶ کلمه از خروجی قبل از استفاده دور انداخته شود.

۴-۵-۳-۳. رمزکنندهٔ جریانی Cisco

رمزکننده‌های جریانی مورد استفاده قرار می‌گیرند به خاطر این حقیقت که:

$$x \text{ XOR } y \text{ XOR } y = x$$

[1] Length-increasing
[2] Secure Hash Algorithm -1
[3] Secure Socket Layer
[4] Session Keys
[5] Initialization Vector

یکی از روش‌های رمزکردن به‌کار گرفته شده توسط مسیریاب‌های Cisco برای رمزکردن کلمه‌های عبور، یک رمزکنندهٔ جریانی است که از یک Keystream ثابت استفاده می‌کند و ۳۸ مدخل نخست آن

```
dsfd;kfoA,.iyewrkldJKDHSUBsgvca69834nc
```

می‌باشد. هنگامی که یک کلمهٔ عبور قرار است رمز شود، تابع کلمهٔ عبور یک عدد بین ۰ تا ۱۵ انتخاب می‌کند که به عنوان آفست Keystream استفاده می‌گردد. سپس کاراکترهای کلمهٔ عبور بر طبق رابطهٔ

$C_i = P_i \text{ XOR } K_{(offset+i)}$

بایت به بایت با Keystream، XOR می‌شوند.

۴-۳-۵-۴. A5

الگوریتم A5 الگوریتم رمزکردنی است که در سیستم‌های GSM مورد استفاده قرار می‌گیرد. پیاده‌سازی‌های مختلفی به نام‌های A5/1، A5/2 و.... موجودند. A5/1 به عنوان الگوریتم قوی Voice-Privacy از طریق هوا شناخته می‌شود. A5/X (A5/2 و....) پیاده‌سازی‌های ضعیف‌تری هستند که به هدف بازارهای خارج اروپا انجام شده‌اند. بررسی‌های انجام شده نشان داده است که علیرغم ادعاهای انجام‌شده این الگوریتم امنیت کاملی را ارائه نمی‌دهد و افراد سودجو می‌توانند از آن طریق به تضییع حقوق استفاده‌کنندگان بپردازند.

۴-۳-۵-۵. Scream

یک رمزکنندهٔ جریانی Software-efficient می‌باشد که همانند SEAL در مرکز تحقیقاتی IBM T.J. Watson طراحی شده است تا امنیتی خیلی بیشتر از SEAL مهیا کند. طراحی Scream از جهات گوناگون به طراحی یک رمزکنندهٔ بلوکی شباهت دارد. رمزکنندهٔ جدید از لحاظ سرعت تقریباً با SEAL هم سرعت است اما از لحاظ امنیت، سطح امنیت بالاتری را ارائه می‌دهد. قابل توجه است که در طراحی این الگوریتم از ایده‌های طراحی Rijndael بخصوص در طراحی S-box های مورد نیاز استفاده شده است.

از لحاظ پیاده‌سازی همانند Rijndael برای پیاده‌سازی در محیط‌های مختلف، مناسب است. مشخصاً پیاده‌سازی در سخت‌افزار و مصالحهٔ[1] مساحت/سرعت در چنین پیاده‌سازیی می‌تواند شبیه Rijndael باشد. به استثنای این که Scream حافظهٔ بیشتری برای جدول پوشش[2] نیاز دارد. همچنین پیاده‌سازی آن برای پردازنده‌های ۸ و ۱۶ بیتی نیز باید خیلی آسان باشد. پر واضح است که Scream برای محیط‌های با حافظهٔ به‌شدت کم مناسب نیست، اما می‌تواند با کمتر از ۴۰۰ بایت حافظه پیاده‌سازی شود اگرچه که چنین پیاده‌سازی خیلی کند خواهد بود.

[1] Trade off
[2] Mask table

۴-۵-۳-۶. SNOW

SNOW یک رمزکنندهٔ جریانی word-oriented است که طراحی آن خیلی ساده است و شامل یک LFSR که از یک ماشین حالت متناهی تغذیه می‌شود، می‌باشد. این الگوریتم توسط Patrik Ekdahl و Tomas Johansson در دانشکدهٔ IT دانشگاه Lund سوئد طراحی شده است.

از اهداف طراحی این رمزکننده ایجاد یک رمزکنندهٔ جریانی خیلی سریع‌تر از AES با هزینه‌های پیاده‌سازی خیلی کمتر در سخت‌افزار و یک سطح امنیت شبیه AES بوده است که این امر تحقق یافته است. بهترین حملهٔ قابل استفاده در مورد این الگوریتم، حمله جستجوی سراسری کلیدها (Brute-Force) می‌باشد که غیرممکن به نظر می‌رسد.

۴-۵-۳-۷. رمزکننده‌های جریانی SOBER t-class (کلاس t)

این رمزکننده توسط Greg Rose و Philip Hawkes در QUALCOMM استرالیا طراحی شده‌است. کلاس‌های مختلف آن t8، t16 و t32 (برای پیاده‌سازی روی پردازنده‌های ۸، ۱۶ و ۳۲ بیتی) به ترتیب قدرت کلید ۶۴، ۱۲۸ و ۲۵۶ بیتی ارائه می‌دهند. رمزکننده‌های کلاس t رمزکننده‌های نرم‌افزاری طراحی‌شده برای پیاده‌سازی نرم‌افزاری می‌باشند. از لحاظ امنیت و پیاده‌سازی خواص خوبی را از خود نشان داده‌اند ولی هنوز اطمینان کاملی دربارهٔ امنیت آنها نیست.

۴-۶. توابع درهم‌سازی

"توابع درهم‌سازی رمزنگار"[1] نقش اساسی در رمزنگاری نوین بازی می‌کنند. این توابع در مقایسه با توابع درهم‌سازی مرسوم مورد استفاده در کاربردهای کامپیوتری غیررمزنگاری- درهر دو حالت، محدوده‌های بزرگتر به نواحی کوچکتر نگاشت می‌شود- درچندین مفهوم مهم متفاوتند.

توابع درهم‌سازی معمولا" یک عنصر مهم از غالب روشهای اعتبارسنجی و امضای دیجیتالی می‌باشند. این توابع یک پیام به طول دلخواه گرفته و یک مقدار با طول ثابت تولید می‌کنند که گاهی اوقات "خلاصهٔ‌پیام"[2] نامیده می‌شود. در این کاربرد نوعا" یک مقدار hash از پیام محاسبه شده و همراه پیام فرستاده می‌شود و در طرف گیرنده، مجدداً hash محاسبه و مقایسه می‌شود. بدین ترتیب، گیرنده قادر به تشخیص "جامعیت داده"[3] خواهد شد. اما امضای یک "خلاصه پیام" به جای خود پیام اغلب بازده و کارآیی پردازش را بهبود می‌بخشد زیرا خلاصهٔ پیام معمولاً از لحاظ اندازه خیلی کوچکتر ازخود پیام است. البته توابع hash مورد استفاده در این کاربرد حتی‌الامکان باید پیامهای ممکن را در تمام محدودهٔ مقادیر hash ممکن به‌نحوی توزیع کند که حدس‌زدن یک پیام که به یک مقدار مشخص hash منجر شود؛ به شدت مشکل باشد.

از دیگر موارد استفادهٔ الگوریتم‌های درهم‌سازی، تولید یک "اثر انگشت دیجیتالی"[4] ازمحتویات یک فایل است که غالبا" به‌جهت اطمینان از عدم تغییر فایل توسط یک نفوذکننده یا ویروس استفاده می‌شود. همچنین سیستمهای عامل برای رمزکردن کلمه‌های عبور از توابع درهم‌سازی استفاده می‌کنند.

قابل ذکر است که هر روش رمزنگاری برای تعدادی کاربردهای خاص بهینه شده است. به عنوان مثال توابع درهم‌سازی مشخصا" مناسب کسب اطمینان از جامعیت داده می‌باشند چراکه هر تغییری در محتوای یک پیام منجر به محاسبهٔ مقدار hash متفاوتی خواهد شد. ازآنجاییکه بسیار غیرمحتمل است که دو پیام متفاوت منجر به یک مقدار hash شوند، جامعیت داده تا حد بالایی از اطمینان تأیید می‌شود.

هرتابع درهم‌سازی قوی می‌تواند در ساختار یک رمزکننده استفاده شود. چندین ترکیب امکان‌پذیر است و ایدهٔ اصلی این است که تابع درهم‌سازی به‌عنوان مولد اعداد تصادفی استفاده شود. در بعضی مواقع برای تولید کلیدهای مورد نیاز نیز از توابع درهم‌سازی استفاده می‌شود.

۴-۶-۱. خصوصیات کلی توابع درهم‌سازی

توابع درهم‌سازی یک پیام را به عنوان ورودی گرفته و یک خروجی تولید می‌کنند که از آن با عنوان hash-value، hash-code، hash-result یا به سادگی hash یاد می‌شود. به بیان دقیق‌تر، تابع درهم‌سازی h، یک رشته‌بیت با طول

[1] Cryptographic hash functions
[2] Message Digest
[3] Data Integrity
[4] Digital Fingerprint

متناهی دلخواه را به رشته‌بیتی با طول ثابت n بیت، نگاشت می‌کند. برای یک دامنهٔ D وبرد R با فرض |D| > |R|، h:D → R تابع "چند به یک"[1] می‌باشدکه به طور ضمنی بیان می‌کند که وجود تصادم[2] (یک جفت ورودی با یک‌خروجی) غیرقابل اجتناب می‌باشد. در واقع، با محدودکردن h به یک دامنه از ورودیهای t بیتی (t >n)، اگر h به‌نحوی "تصادفی" باشد که تمام خروجیها اساسا" با یک احتمال تولید شوند، آنگاه حدود ۲^{t-n} ورودی به هر خروجی نگاشته خواهد شد، و دو ورودی تصادفی انتخاب‌شده به یک خروجی با احتمال ۲^{-n} (مستقل از t) منجر خواهند شد. ایدهٔ اصلی توابع درهم‌سازی رمزنگار این است که یک مقدار hash به عنوان یک "تصویرفشرده"[3] (گاهی اوقات به نام نشان[4]، اثرانگشت دیجیتالی یا خلاصه پیام) از رشتهٔ ورودی عمل می‌کند و با آن رشته به طور منحصربفرد قابل شناسایی است.

توابع درهم‌سازی به‌همراه روشهای امضای دیجیتالی برای جامعیت داده مورد استفاد قرار می‌گیرند؛ جاییکه به چندین دلیل نوعا" یک پیام، نخست hash می‌شود و سپس مقدار hash، به عنوان یک نمایش از پیام، به جای پیام اصلی امضاء می‌شود. یک دستهٔ مجزا از توابع hash، به نام "کدهای اعتبارسنجی پیام" (MACs[5]) بوسیلهٔ تکنیک‌های متقارن، اعتبارسنجی پیام را ممکن می‌کنند. الگوریتم‌هایMAC می‌توانند به‌عنوان توابع hash ی در نظرگرفته شوند که عملاً دو ورودی مجزا، یک پیام و یک کلید محرمانه را گرفته و یک خروجی با اندازهٔ ثابت تولید می‌کنند با این هدف که درعمل تولید همان خروجی بدون دانستن کلید غیرممکن باشد. MAC ها می‌توانند به جهت مهیا کردن جامعیت داده و اعتبارسنجی متقارن منبع داده همانند شناسایی[6] در روشهای کلید متقارن مورد استفاده قرارگیرند.

یک استفادهٔ نوعی از توابع درهم‌سازی (بدون کلید) برای جامعیت داده می‌تواند مانند زیر باشد. مقدار hash متناظر با یک پیام مشخص x در زمان T_1 محاسبه می‌شود. جامعیت این مقدار hash (نه خود پیام) به نحوی محافظت می‌شود. درزمان بعدی T2، آزمایش زیر جهت تشخیص تغییر پیام انجام می‌شود؛ یعنی آیا پیام x´ همانند پیام اصلی است . مقدار hash x´ محاسبه می‌شود و با مقدار hash محافظت شده مقایسه می‌گردد. اگرآنها یکسان باشند، استفاده‌کننده می‌پذیرد که ورودیها یکسان هستند و بنابراین پیام تغییر نیافته است. بدین ترتیب مشکل حفظ جامعیت یک پیام بالقوه بزرگ به حفظ جامعیت یک مقدار hash کوچک کاهش می‌یابد. از آنجاییکه وجود تصادم در نگاشتهای "چند به یک" حتمی است ارتباط یگانهٔ بین ورودیها و مقادیر hash در بهترین حالت مربوط به مفهوم محاسباتی بودن می‌باشد. در عمل یک مقدار hash باید بطور یگانه با یک ورودی قابل تشخیص باشد و بطور محاسباتی[7]، یافتن تصادمها باید مشکل باشد (اساسا" در عمل نباید رخ دهند).

[1] Many-to-one
[2] Collision
[3] Compact image
[4] Imprint
[5] Message Authentication Codes
[6] Identification
[7] Computationally

۴-۶-۲. دسته بندی توابع درهم سازی

دربالاترین سطح، توابع درهم‌سازی ممکن است به دو دسته (کلاس) تقسیم شوند:

- "توابع درهم‌سازی بدون کلید"[1] که یک پارامتر ورودی (یک پیام) می‌پذیرند.
- توابع درهم‌سازی کلیددار که دو ورودی مجزا می‌پذیرند؛ یک پیام و یک کلید محرمانه.

برای سادگی بحث یک تابع درهم‌سازی بطور غیر رسمی به صورت زیر تعریف می‌شود:

یک تابع hash (درمفهوم کلی) یک تابع h است که حداقل دو خاصیت زیر را داراست :

- فشردگی[2] - h یک ورودی x با طول بیت متناهی دلخواه، را به یک خروجی h(x) با طول بیت ثابت n نگاشت می کند.
- سادگی محاسبه[3]- با داشتن h و یک ورودی x، h(x) به سادگی محاسبه می‌شود.

با تعریف داده‌شده، تابع درهم‌سازی به تابع درهم‌سازی بدون کلید اشاره می‌کند. در زمانی که بحث درسطح کلی است، ممکن است این ترکیب قدری ابهام ایجاد کندکه بوسیلهٔ توضیح در متن قابل برطرف شدن می‌باشد.

برای استفادهٔ واقعی، یک دسته‌بندی مبتنی برهدف از توابع درهم‌سازی (ماورای کلیددار در برابر بدون کلید) براساس خواص دیگری که آنها مهیا می‌کنند و نیازهای منتج شده از کاربردهای مشخص، لازم است. ازدسته‌بندیهای بیشمار از این نوع، به دو نوع توابع درهم‌سازی اشاره می‌شود:

- *کدهای تشخیص تغییر (MDCs[4])*

این گروه همچنین به نام "کدهای تشخیص دستکاری"[5] یا کدهای جامعیت پیام (MICs) نیز مشهورند. هدف از یک MDC (بصورت غیررسمی) مهیا کردن یک تصویر نمایشگر یا hash از پیام به منظور برآوردن بعضی خواص اضافی است. هدف نهایی، تسهیل اطمینان ازجامعیت داده، به‌همراه مکانیزمهای اضافی موردنیاز با کاربرد خاص می‌باشد. MDC ها یک زیربخش از توابع درهم‌سازی بدون کلید هستند و خودشان ممکن است بیشتر دسته‌بندی شوند؛ کلاسهای مشخص MDC ها که در این گزارش به آنها پرداخته شده‌است عبارتند از:

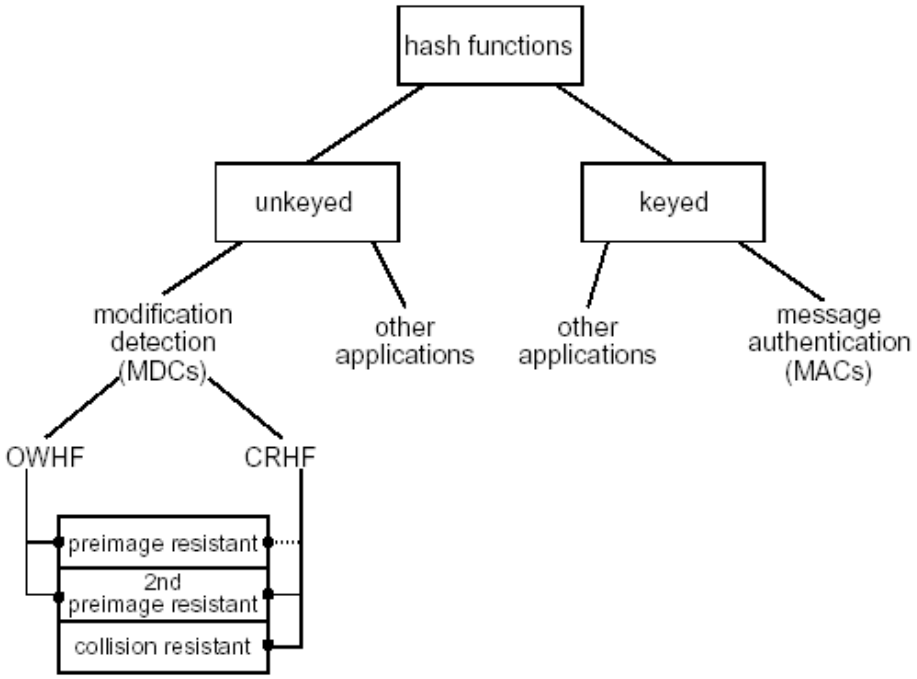
- توابع درهم‌سازی یک‌طرفه (OWHFs[6]) : برای این دسته، یافتن یک ورودی که به یک مقدار hash از پیش مشخص شده hash شود، مشکل است.

[1] Unkeyed hash functions
[2] Compression
[3] Ease of computation
[4] Modification detection codes
[5] Manipulation detection codes
[6] One-Way Hash Functions

- توابع درهم‌سازی مقاوم دربرابر تصادم (CRHFs[1]): برای این دسته، یافتن دو ورودی که یک مقدار hash داشته باشند مشکل است.

- *کدهای اعتبارسنجی پیام (MACs[2])*

هدف از یک MAC (بصورت غیررسمی) تسهیل اطمینان یافتن هم از منبع یک پیام و هم از جامعیت آن بدون استفاده از هرمکانیزم اضافی می‌باشد. MAC ها عملا" دو پارامتر مجزا دارند، یک پیام ورودی ویک کلیدمحرمانه؛ آنها یک زیربخش از توابع درهم‌سازی کلیددار می‌باشند. شکل ۴-۲۲ این دسته‌بندی ساده شده را بیان می کند.



شکل ۴-۲۲ دسته‌بندی ساده‌شدهٔ توابع درهم‌سازی رمزنگار و کاربرد آنها

۴-۶-۳. کاربردهای دیگر توابع درهم‌سازی بدون کلید

کاربردهای دیگر توابع درهم‌سازی بدون‌کلید همان‌طورکه در مقدمه نیز آمده است عبارتند از :

- "تأیید آگاهی"[3]: کسب اطمینان ازیک آگاهی بدون فاش شدن خود آگاهی؛ مانند مکانیزم مورد استفاده برای کلمه‌های عبور، کلمه عبور عینا" درجایی ذخیره نمی‌گردد بلکه تنها یک مقدار hash از آن نگهداری می‌شود که درصورت لزوم با مقدار hash کلمه عبور وارد شده بوسیلهٔ استفاده‌کننده مقایسه می‌شود.

[1] Collision Resistant Hash Functions
[2] Message Authentication Codes
[3] Confirmation of knowledge

- "اشتقاق کلید"[1]: محاسبهٔ دنباله‌های کلیدهای جدید از کلیدهای قبلی. یک مثال اولیه اشتقاق کلید درپایانه‌های PoS[2] می‌باشد؛ در اینجا یک نیاز مهم این است که مصالحه (فاش شدن) کلیدهای قبلی مبادله[3] نباید باعث مصالحهٔ امنیت کلیدهای فعال فعلی شود. نمونهٔ دیگر تولید دنباله‌های one-time password براساس توابع یکطرفه می‌باشد.

- تولید اعداد شبه‌تصادفی: برای تولید دنباله‌هایی از اعداد که خواص تصادفی دارند (یک مولد اعداد شبه‌تصادفی می‌تواند برای ایجاد یک رمزکنندهٔ بلوکی کلیدمتقارن استفاده شود). به دلیل مشکل‌بودن تولید اعداد شبه تصادفی قوی از نظر رمزنگاری، MDCها نباید برای این هدف استفاده شوند مگراینکه نیازهای تصادفی بودن به‌خوبی مطالعه شده باشند و MDC برای برآوردن آن اهداف اصلاح شود.

نکته‌ای که دراستفاده از MDCها حائز اهمیت است این است‌که تعداد زیادی ازMDCها که درعمل مورد استفاده قرار می‌گیرند ممکن است این‌طور به نظر برسد که نیازهای اضافی ماورای آنهایی را که برایشان طراحی شده‌اند برآورده می‌کنند. با این حال، استفاده از توابع درهم‌سازی دلخواه برای هرکاربرد بدون بررسی وتحلیلی که دقیقا" هم خواص حیاتی مورد نیاز کاربرد و هم آنهایی را که بوسیلهٔ تابع برآورده می‌شود را مشخص می‌کند، نمی‌تواند توصیه شود.

۴-۶-۴. خواص اضافی توابع درهم‌سازی یکطرفه

خواص اضافی توابع درهم‌سازی یک‌طرفه مورد انتظار برای کاربردهای بیان‌شده درقبل شامل موارد زیر می‌باشد :

- نابستگی[4]. بیتهای ورودی و خروجی نباید وابستگی داشته باشند. بنابراین یک خاصیت بهمنی[5] شبیه رمزکننده‌های بلوکی خوب مطلوب است که به موجب آن هر بیت ورودی همه بیتهای خروجی را متأثر می‌کند.

- مقاومت near-collision. باید یافتن هردو ورودی x و x´ که h(x) و h(x´) فقط در تعداد کمی از بیتها متفاوت باشند، سخت باشد.

- مقاومت partial-preimage یا "یکطرفه بودن محلی"[6]. باید بازیافت هر زیررشته به مشکلی بازیافت همهٔ ورودی باشد. بعلاوه، حتی اگر بخشی از ورودی معلوم باشد، باید پیداکردن بقیهٔ ورودی مشکل باشد (یعنی اگر t بیت ورودی نامعلوم باشند، باید به طور متوسط 2^{t-1} عمل hash برای یافتن این بیت‌ها صورت گیرد).

کاربردهای دیگر توابع درهم‌سازی کلیددار شامل استفاده در پروتکل‌های شناساییchallenge- response برای پردازش پاسخهایی که تابعی از یک کلید محرمانه و یک challenge message می‌باشند و برای "تصدیق کلید"[1] می‌باشد. باید بین یک الگوریتم MAC و استفاده از یک MDC با یک کلید محرمانه بعنوان بخشی از پیام ورودی‌اش تفاوت قائل شد.

[1] Key derivation
[2] Point -of-Sale
[3] Transaction
[4] Non-correlation
[5] Avalanche
[6] Local one-wayness

به طورکلی فرض می‌شود که توصیف الگوریتمی یک تابع درهم‌سازی دانشی عمومی است. بنابراین درحالت MDC ، با داشتن یک پیام به عنوان ورودی، هرکسی می‌تواند hash-result را محاسبه کند، و در حالت MAC ها و با داشتن یک پیام به عنوان ورودی ، هرکس با دانستن کلید می‌تواند hash-result را محاسبه کند.

علاوه بر فشردگی و سادگی محاسبه که قبلا" بیان شده است برای یک تابع درهم سازی بدون کلید h با ورودیهای x و x´ و خروجیهای y و y´ خواص زیر لیست می‌شوند:

- مقاومت preimage : اساسا" برای هر خروجی ازپیش مشخص‌شده، بصورت محاسباتی یافتن هرورودی که به آن خروجی hash شود غیرممکن است؛ یعنی، پیداکردن هر preimage x´ که h(x´) = y با داشتن هر y که ورودی متناظر آن نامعلوم است، غیرممکن می‌باشد.
- مقاومت 2nd- preimage : بصورت محاسباتی پیداکردن هرورودی دومی که همان خروجی رابرای یک ورودی مشخص‌شده داشته باشد؛ غیرممکن است یعنی با داشتن x، پیداکردن 2nd-preimage x´ که مخالف x است به‌نحوی‌که h(x) = h(x´)، غیرممکن می‌باشد.
- "مقاومت دربرابر تصادم"[2]: بصورت محاسباتی یافتن هردو ورودی مجزای x و x´ که به یک مقدار hash می‌شوند؛ غیرممکن است یعنی؛ به‌نحوی‌که h(x)= h(x´).

البته دقت شود که منظور از سادگی در محاسبات به معنی زمان و فضای خطی (چندجمله‌ای) یا خیلی عملی، بایک مقدار مشخص از عملیات ماشین یا واحدهای زمانی قابل انجام‌شدن می‌باشد.

برای هرکاربرد مشخص از توابع درهم‌سازی خواص مشخصی نیز باید کسب شودکه درجدول ۴-۱۲ این خواص و کاربردها دیده می‌شوند.

جدول ۴-۱۲ خواص مقاومتی مورد نیاز برای انواع کاربردهای جامعیت داده

Hash properties required → Integrity application ↓	Preimage resistant	2nd-preimage	Collision resistant
MDC + asymmetric signature	yes	yes	yes†
MDC + authentic channel		yes	yes†
MDC + symmetric encryption			
hash for one-way password file	yes		
MAC (key unknown to attacker)	yes	yes	yes†
MAC (key known to attacker)		yes‡	

[1] Key confirmation
[2] Collision resistance

۴-۶-۵. توابع درهم‌سازی مهم

دراین گروه از الگوریتم‌های رمزنگاری نیز درهرزمان استانداردهایی معرفی شده‌اند که معروفترین آنها خانوادهٔ MD[1] (MD2، MD4 و MD5) و SHS[2] می‌باشند که البته MD2 و MD4 شکسته شده‌اند و گفته می‌شود که MD5 نیز دربرابر "حملات تصادم"[3] آسیب‌پذیر است اگرچه که به طور کامل شکسته نشده است. شرکت RSA Data Security که خود ایجادکنندهٔ این سری از الگوریتم‌ها می‌باشد استفاده از MD5 را برای کاربردهایی که نیازمند مقاوم‌بودن در برابر حملات تصادمی هستند مناسب نمی‌داند. اما استاندارد SHS که در FIPS PUB 180-2 توسط NIST معرفی شده‌است تنها استانداردی است که فعلاً بدون مشکل دانسته می‌شود. دراین استاندارد از الگوریتم‌های SHA-1 ، SHA-256، SHA-384 و SHA-512 استفاده می‌شود.

در ادامه به شرح سری الگوریتم‌های SHA، Whirlpool (الگوریتم نهایی بخش توابع درهم‌سازی ارائه‌شده در پروژهٔ NESSIE)، RIPEMD-160 و Tiger به عنوان الگوریتم‌های جدیدتر (موردبحث امروزه) خواهیم پرداخت و سپس اطلاعات مختصری دربارهٔ تعدادی از الگوریتمهای منتخب که تا حدی مورد استفاده بوده‌اند و یا زمانی پرکاربرد بوده‌اند ارائه خواهد شد.

۴-۶-۵-۱. استاندارد SHS (گاهی اوقات به نام SHA نیز گفته می‌شود)

- **خواص کلی توابع درهم‌سازی سری SHA**

همان‌طورکه قبلا" ذکر شد دراین استاندارد استفاده از الگوریتم‌های SHA-1، SHA-256، SHA-384 و SHA-512 پیشنهاد می‌شود که الگوریتم‌های SHA-256، SHA-384 و SHA-512 به‌تازگی به استاندارد قبلی اضافه شده‌اند (قبلا" تنها SHA-1 در FIPS PUB 180-1 معرفی شده بود).

همهٔ این چهار الگوریتم توابع تکراری یکطرفهٔ درهم‌سازی هستند که می‌توانند یک پیام را برای تولید یک نمایش فشرده به نام "خلاصه پیام" پردازش کنند.

هر الگوریتم می‌تواند در دو مرحله تشریح شود: پیش‌پردازش و محاسبهٔ hash. مرحلهٔ پیش‌پردازش شامل padding پیام و تجزیهٔ پیام pad شده به بلوکهای m بیتی و نشاندن مقادیر اولیهٔ مورد استفاده درمحاسبهٔ hash می‌باشد. درمرحلهٔ محاسبهٔ hash یک "زمانبندی پیام"[4] از پیام pad شده تولید شده و آن زمانبندی به‌همراه توابع، ثابتها و عملیات کلمه‌ای برای تولید تکراری یک مجموعه از مقادیر hash مورد استفاده قرار می‌گیرد. مقدار hash نهایی تولید شده بوسیلهٔ این مرحله به عنوان نتیجه، استفاده می‌گردد.

[1] Message Digest
[2] Secure Hash Signature Standard
[3] Collision attack
[4] Message schedule

چهار الگوریتم نام برده شده به طرز قابل توجهی درتعداد بیتهای امنیت برای داده‌ای که قرار است hash شود، فرق می‌کنند که مستقیماً مربوط به طول خلاصه پیام است. زمانیکه یک الگوریتم hash امن (SHA) - امن به این دلیل که ازلحاظ محاسباتی یافتن یک پیام متناظر با یک خلاصه پیام داده‌شده یا پیداکردن دو پیام مختلف که یک خلاصه پیام تولید کنند، غیرممکن است - به همراه الگوریتم دیگری مورد استفاده قرار می‌گیرد، درچنین زمانی ممکن است برای تطبیق، به تعداد مشخصی از بیتهای امنیت نیاز باشد. به عنوان مثال اگر یک پیام بوسیلهٔ یک الگوریتم امضای دیجیتالی امضاء شود که ۱۲۸ بیت امنیت را مهیا می‌کند ، همچنین به یک الگوریتم درهم‌سازی امن که ۱۲۸ بیت امنیت را برآورده می‌کند (مثلاً SHA-256) نیاز می‌باشد. SHA-1 یک خلاصه پیام ۱۶۰ بیتی تولید می‌کند که بیشتر از ۸۰ بیت امنیت دربرابر حملات تصادم فراهم نمی‌کند. از طرفی AES سه اندازهٔ کلید پیشنهاد می‌دهد : ۱۲۸، ۱۹۲ و ۲۵۶ بیتی. برای یک الگوریتم درهم‌سازی همراه، نیاز است که سطوح مشابهی از امنیت را مهیا کند. علاوه بر این، چهار الگوریتم در ترمهایی مانند اندازهٔ بلوکها و کلمات داده‌ای که درخلال درهم‌سازی استفاده می‌شوند نیز متفاوتند. جدول ۴-۱۳ خصوصیات اساسی این الگوریتم‌ها را نشان می‌دهد:

جدول ۴-۱۳ خصوصیات اساسی الگوریتم‌های درهم‌سازی امن (SHAs)

Algorithm	Message Size (bits)	Block Size (bits)	Word Size (bits)	Message Digest Size (bits)	Security[2] (bits)
SHA-1	$< 2^{64}$	512	32	160	80
SHA-256	$< 2^{64}$	512	32	256	128
SHA-384	$< 2^{128}$	1024	64	384	192
SHA-512	$< 2^{128}$	1024	64	512	256

◆ SHA-1

SHA-1 می‌تواند برای hash نمودن پیام M به طول l بیت که $0 <= l < 2^{64}$ استفاده شود. الگوریتم از ۱) یک زمانبندی پیام از ۸۰ کلمهٔ ۳۲ بیتی، ۲) ۵ متغیرکاری ۳۲ بیتی و ۳) یک مقدار hash از پنج کلمهٔ ۳۲ بیتی استفاده می‌کند. نتیجهٔ نهایی SHA-1 یک خلاصه پیام ۱۶۰بیتی می‌باشد.

کلمات زمانبندی پیام برچسبهای W_0، W_1، ... و W_{79} را دارند. پنج متغیرکاری a، b، c، d و e می‌باشند. کلمات حاوی مقادیر hash هم $H_0^{(i)}$، $H_1^{(i)}$، ... و $H_4^{(i)}$ نام دارند که در ابتدا مقدار اولیهٔ hash، $H^{(0)}$ را نگهداری می‌کنند و (بعد از پردازش هر بلوک پیام) بوسیلهٔ مقدار hash میانی بعدی ($H^{(i)}$) جایگزین خواهد شد و سرانجام با مقدارنهایی hash، $H^{(N)}$ پایان می‌پذیرند. SHA-1 همچنین از یک کلمهٔ موقتی به نام T استفاده می‌کند.

- ***پیش پردازش SHA-1***

۱. پیام M؛ pad می‌شود. (تعدادی `0` و `1` به جهت اینکه طول پیام pad شده مضربی از ۵۱۲ بیت باشد، به انتهای آن اضافه می‌شود). برای SHA-256 نیز همین عمل دقیقا" انجام می‌شود.

۲. پیام pad شده به N بلوک پیام ۵۱۲ بیتی تقسیم می‌شود ($M^{(1)}$، $M^{(2)}$، ... و $M^{(N)}$). از آنجاییکه ۵۱۲ بیت بلوک ورودی می‌تواند به‌عنوان ۱۶ کلمهٔ ۳۲ بیتی بیان شود، اولین ۳۲ بیت از بلوک i پیام با $M_0^{(i)}$، ۳۲ بیت بعدی $M_1^{(i)}$ و به همین ترتیب تا $M_{15}^{(i)}$ نمایش داده می‌شود. (همین‌طور برای SHA-256).

۳. مقدار اولیهٔ hash نشانده می‌شود ($H^{(0)}$).

برای SHA-1، مقدار اولیهٔ hash، $H^{(0)}$ شامل پنج کلمهٔ ۳۲ بیتی زیر (در مبنای ۱۶) می‌باشد.

$$
\begin{aligned}
H_0^{(0)} &= 67452301 \\
H_1^{(0)} &= efcdab89 \\
H_2^{(0)} &= 98badcfe \\
H_3^{(0)} &= 10325476 \\
H_4^{(0)} &= c3d2e1f0
\end{aligned}
$$

برای SHA-256، هشت کلمهٔ ۳۲ بیتی وجود دارد:

$$
\begin{aligned}
H_0^{(0)} &= 6a09e667 \\
H_1^{(0)} &= bb67ae85 \\
H_2^{(0)} &= 3c6ef372 \\
H_3^{(0)} &= a54ff53a \\
H_4^{(0)} &= 510e527f \\
H_5^{(0)} &= 9b05688c \\
H_6^{(0)} &= 1f83d9ab \\
H_7^{(0)} &= 5be0cd19.
\end{aligned}
$$

نحوهٔ محاسبهٔ hash در SHA-1 به صورت زیر است:

- توابع SHA-1 : SHA-1 ازیک توالی توابع منطقی f_0، f_1، ... و f_{79} استفاده می‌کند. هر تابع f_t که در آن 0<= t <= 79، روی سه کلمهٔ ۳۲ بیتی x، y و z عمل می‌کند و یک کلمهٔ ۳۲ بیتی به عنوان خروجی تولید می‌کند. تابع $f_t(x,y,z)$ مطابق زیر تعریف می‌شود :

$$
f_t(x,y,z) = \begin{cases} (x \wedge y) \vee (\neg x \wedge z) & 0 \le t \le 19 \\ x \oplus y \oplus z & 20 \le t \le 39 \\ (x \wedge y) \vee (x \wedge z) \vee (y \wedge z) & 40 \le t \le 59 \\ x \oplus y \oplus z & 60 \le t \le 79. \end{cases}
$$

- ثابتهای SHA-1 : SHA-1 از یک توالی متشکل از ۸۰ کلمهٔ ۳۲ بیتی ثابت K_0، K_1، ... و K_{79} استفاده می‌کند که عبارتند از:

$$K_t = \begin{cases} \texttt{5a827999} & 0 \le t \le 19 \\ \texttt{6ed9eba1} & 20 \le t \le 39 \\ \texttt{8f1bbcdc} & 40 \le t \le 59 \\ \texttt{ca62c1d6} & 60 \le t \le 79. \end{cases}$$

بعد از اینکه پیش‌پردازش تکمیل شد، هر بلوک پیام، $M^{(1)}$، $M^{(2)}$، ... و $M^{(N)}$ به ترتیب با استفاده از گامهای زیر (شکل ۴-۲۳) پردازش می‌شوند: (ابتدا به توضیح بعضی علائم پرداخته می‌شود)

علائم و عملیات استفاده شده در گامهای توضیح داده شده از قرار زیرند:

- $\oplus$: یای انحصاری (XOR)
- جمع (+) به پیمانهٔ 2^{32} انجام می‌شود.
- $\text{ROTL}^n(x) = (x << n) \vee (x >> w - n)$ (چرخش به چپ x به تعداد n)

For $i = 1$ to N:

{

1. Prepare the message schedule, $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

2. Initialize the five working variables, ***a***, ***b***, ***c***, ***d***, and ***e***, with the $(i\text{-}1)^{st}$ hash value:

$$a = H_0^{(i)}$$
$$b = H_1^{(i)}$$
$$c = H_2^{(i)}$$
$$d = H_3^{(i)}$$
$$e = H_4^{(i)}$$

3. For $t = 0$ to 79:

{

$$T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_t$$
$$e = d$$
$$d = c$$
$$c = ROTL^{30}(b)$$
$$b = a$$
$$a = T$$

}

4. Compute the i^{th} intermediate hash value $H^{(i)}$:

$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$

}

شکل ۴- ۲۳ گامهای الگوریتم SHA-1

بعد از تکرار گامهای ۱ تا ۴، به تعداد N بار (یعنی پس از پردازش $M^{(N)}$)، خلاصه پیام ۱۶۰ بیتی حاصل از پیام M توالی $H_0^{(N)} || H_1^{(N)} || H_2^{(N)} || H_3^{(N)} || H_4^{(N)}$ خواهد بود.

البته این روش محاسبهٔ hash فرض می‌کند که W_0، W_1، ... و W_{79} به عنوان یک آرایه از ۸۰ کلمهٔ ۳۲ بیتی پیاده سازی شده است که از دیدگاه حداقل‌کردن زمان اجرا کارا است، چراکه آدرسهای W_{t-3}،

W_{t-4}، ... و W_{t-16} درگام ۱ از الگوریتم قبل به‌آسانی محاسبه می‌شوند. به هرحال اگر حافظه محدود باشد، یک راه حل جایگزین درنظرگرفتن $\{W_t\}$ به‌عنوان یک صف ورودی است که ممکن است با استفاده از یک آرایه از ۱۶ کلمهٔ ۳۲ بیتی W_0، W_1، ... ، W_{15} پیاده سازی شود. این روش نیز همان خلاصه پیام را تولید می‌کند. علیرغم اینکه روش جدید ۶۴ کلمهٔ ۳۲ بیتی حافظه را صرفه‌جویی می‌کند، بخاطر افزایش یافتن پیچیدگی محاسبهٔ آدرس برای $\{W_t\}$ درگام ۳ باعث افزایش زمان اجرا می‌شود.

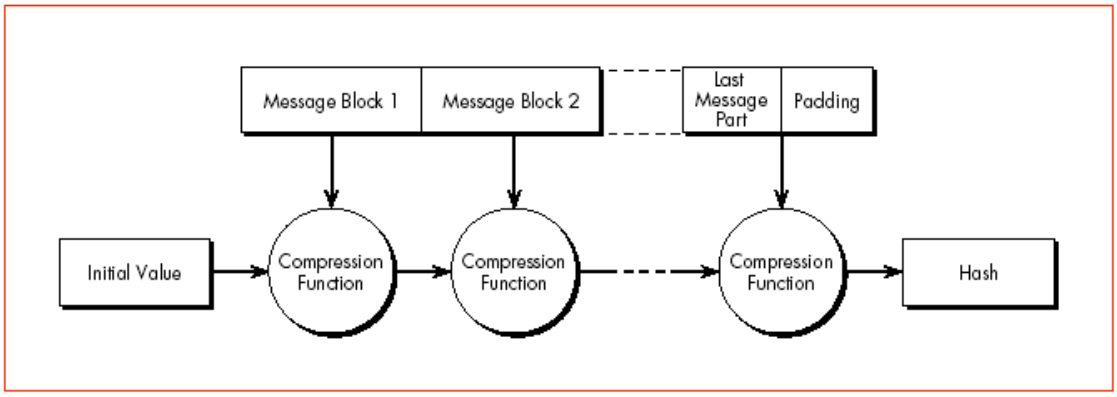
◆ سایر الگوریتم‌های سری SHA

سایر الگوریتم‌های این سری نیز تنها در تعداد ثابتها، مقدار آنها، اعمال و توابع مورد استفاده با هم تفاوت دارند و ساختار اصلی این چهار الگوریتم یکی است. البته SHA-384 دقیقا" مشابه SHA-512 می‌باشد غیر از اینکه پس از پایان عملیات، سمت چپ‌ترین ۳۸۴ بیت خروجی به عنوان نتیجهٔ hash استخراج می‌شود.

نکتهٔ جالب دراین الگوریتم‌ها انتخاب ثابتهای استفاده شده است که درزیر خلاصه شده‌اند:

- SHA-256 : ۸ کلمهٔ ۳۲ بیتی که اولین ۳۲ بیت از قسمت اعشاری ریشهٔ دوم اولین ۸ عدد اول هستند.
- SHA-384 : ۸ کلمهٔ ۶۴ بیتی که اولین ۶۴ بیت از قسمت اعشاری ریشهٔ دوم اعداد اول نهم تا شانزدهم می‌باشند.
- SHA-512 : ۸ کلمهٔ ۶۴ بیتی که اولین ۶۴ بیت از قسمت اعشاری ریشهٔ دوم اولین ۸ عدد اول هستند.

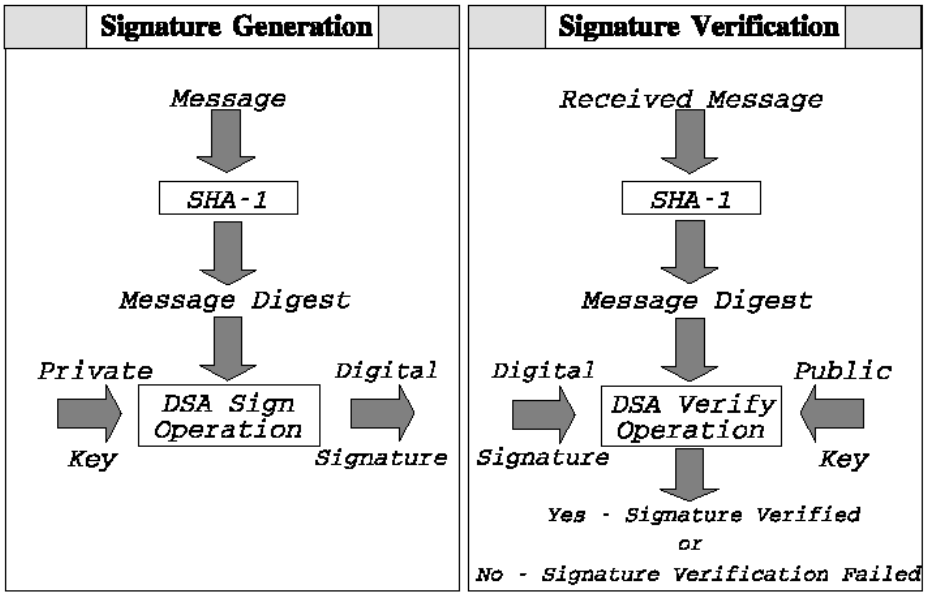
SHA-1 یک تجدید نظر فنی از SHA (FIPS 180) می‌باشد که درآن یک عمل جابجایی به چپ دوری در محلهایی از پردازش اضافه شد. این تجدید نظر، امنیت تأمین شده بوسیلهٔ این استاندارد را بهبود می‌بخشد. SHA-1 بر پایهٔ قواعدی شبیه به آنهایی که بوسیلهٔ پروفسور رونالد ریوست از MIT در طراحی MD4 استفاده شد، بنا شده است و یک طرح بسیار نزدیک به آن می‌باشد. ساختار کلی این توابع در شکل ۴-۲۴ نشان داده شده است.



شکل ۴-۲۴ استفاده از تابع فشرده‌سازی در یک تابع درهم‌سازی تکراری. توابع درهم‌سازی MD2، MD4 و SHA-1 اساساً این طراحی را دنبال می‌کنند.

◆ کاربردها

SHA-1 ممکن است به‌همراه DSA[1] درنامهٔ الکترونیکی، تبادل پول الکترونیکی، توزیع نرم‌افزار، ذخیرهٔ داده و دیگر کاربردهایی که به اطمینان از جامعیت داده و تأیید مرجع داده نیاز است، استفاده شود (شکل ۴-۲۵).



شکل ۴-۲۵ استفاده از SHA-1 به‌همراه DSA

◆ پیاده‌سازی‌ها

SHA-1 ممکن است در نرم‌افزار، firmware، سخت‌افزار و یا ترکیبی از اینها پیاده‌سازی شود. به هرحال تنها پیاده‌سازی‌هایی بوسیلهٔ NIST تأیید می‌شوند که از استاندارد پیروی کنند.

نکته : پیروی از این استاندارد برای امن بودن یک پیاده‌سازی خاص کافی نیست. این استاندارد هر پنج سال برای تشخیص کفایتش بازبینی می‌شود.

۴-۶-۵-۲. تابع درهم‌سازی Whirlpool

Whirlpool یک تابع درهم‌سازی است که بوسیلهٔ Vincent Rijmen و Paulo S. L. M. Barreto طراحی شده است و روی پیام‌های کوچکتر از 2^{256} بیت عمل می‌کند و یک خلاصه‌پیام ۵۱۲ بیتی تولید می‌کند. این الگوریتم به عنوان یک الگوریتم کاندید برای پروژهٔ NESSIE اتحادیهٔ اروپا ارائه شده است و به عنوان فینالیست هم انتخاب گردیده است. سازمان

[1] Digital Signature Algorithm

بین المللی استانداردها (ISO [1]) اخیراً تصمیم گرفته است که نسخهٔ tweaked این الگوریتم را در تجدید نظر بعدی استاندارد ISO/IEC 10118 وارد کند.

◆ توضیح تابع

Whirlpool از روش تقویت Merkle-Damgard و روش درهم‌سازی Miyaguchi-Preneel با یک رمزکنندهٔ بلوکی اختصاصی ۵۱۲ بیتی به نام W استفاده می‌کند. رشته بیتی که می‌خواهد hash شود بوسیلهٔ یک بیت `1` و سپس با یک توالی از بیتهای `0` و نهایتا" با طول اولیه (به شکل یک مقدار صحیح ۲۵۶ بیتی) بسط داده می‌شود (Padding)؛ به‌نحوی‌که طول پیام بعد از این عمل مضربی از ۵۱۲ بیت می‌شود. رشته پیام حاصل شده به توالی از بلوکهای ۵۱۲ بیتی m_1، m_2، ... ، m_t تقسیم می‌شود که از آنها برای تولید توالی مقادیر hash H_0، H_1، ... ، H_t استفاده می‌شود. طبق تعریف، H_0 یک رشتهٔ ۵۱۲ بیتی از `0` ها می‌باشد. برای محاسبهٔ H_i، m_i بوسیلهٔ رمزکنندهٔ W با استفاده از H_{i-1} به عنوان کلید، رمز می‌شود و ciphertext حاصل با H_{i-1} و m_i ، XOR می‌شود. سرانجام، H_t خروجی Whirlpool می‌باشد.

◆ رمزکنندهٔ بلوکی W

رمزکنندهٔ بلوکی W استفاده شده توسط Whirlpool خیلی شبیه الگوریتم برندهٔ AES ، Rijndael می‌باشد. تفاوتهای اصلی در جدول ۴-۱۴ خلاصه شده‌اند.

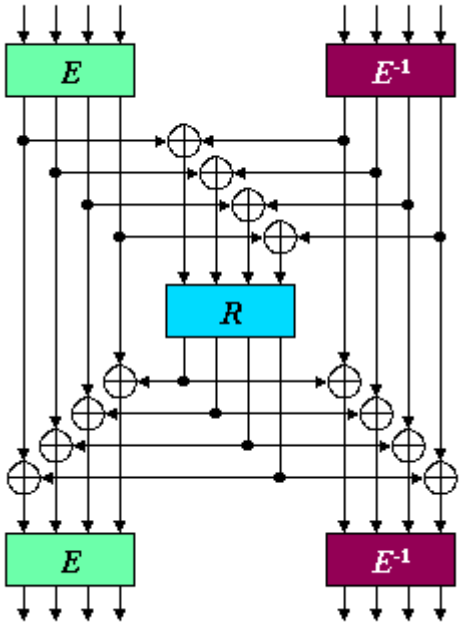
جدول ۴-۱۴ تفاوتهای میان Rijndael و W

	Rijndael	W
Block size (bits)	128, 192, or 256	always 512
Number of rounds	10, 12, or 14	always 10
Key schedule	dedicated *a priori* algorithm	the round function itself
GF(2^8) reduction polynomial	$x^8 + x^4 + x^3 + x + 1$ (0x11B)	$X^8 + x^4 + x^3 + x^2 + 1$ (0x11D)
Origin of the S-box	mapping u -> u^{-1} over GF(2^8), plus affine transform	pseudo-random table (tweak: recursive structure)
Origin of the round constants	polynomials x^i over GF(2^8)	successive entries of the S-box

اخیراً ایجادکنندگان Whirlpool گونهٔ tweaked این الگوریتم را پیشنهاد دادند که آن را برای پیاده‌سازی سخت‌افزاری کاراتر می‌کند. این تغییر، ساختار زیربنایی Whirlpool را تغییر نمی‌دهد، بلکه S-box ی را جایگزین می‌کند که درنسخهٔ اصلی کاملاً تصادفی تولید می‌شود (یعنی فاقد هرگونه ساختار داخلی است). S-box جدید ۸*۸ که بوسیلهٔ یک ساختار

[1] International Standard Organization

بازگشتی تولید می‌شود از mini-box های کوچکتر ۴*۴ تشکیل شده است که دو تا از آنها (E-box و معکوسش) از تابع نمایی روی GF(24) مشتق می‌شوند ویکی (R-box) به صورت شبه تصادفی تولید می‌شود. این ساختار و mini-box ها در شکلهای ۴-۲۶ تا ۴-۲۸ نشان داده شده‌اند.



شکل ۴- ۲۶ ساختار بازگشتی S-box tweaked شده

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E(u)	1	B	9	C	D	6	F	3	E	8	7	4	A	2	5	0

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$E^{-1}(u)$	F	0	D	7	B	E	5	A	9	2	C	1	3	4	8	6

شکل ۴-۲۷ mini-box های E و E^{-1}

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
R(u)	7	C	B	D	E	4	9	F	6	3	8	A	2	5	1	0

شکل ۴- ۲۸ R mini-box

◆ امنیت Whirlpool

برگزیده‌ای از مستندات ارائه شده به NESSIE در زیرآمده است:

بافرض اینکه اگر مقدار هرزیررشتهٔ n بیتی از خروجی کامل Whirlpool به عنوان مقدار hash برداشته شود، طراحی Whirlpool اهداف امنیتی زیرین را برآورده می کند:

- مقدار عمل لازم و مورد انتظار برای تولید یک تصادم از مرتبهٔ $۲^{n/۲}$ اجرای Whirlpool می‌باشد.
- مقدارعمل مورد نیاز برای پیداکردن یک پیام که به یک مقدار داده شدهٔ n بیتی hash شود از مرتبهٔ ۲^{n} اجرای Whirlpool می‌باشد.
- یک پیام و نتیجهٔ hash n بیتی آن داده شده‌اند؛ مقدار عمل مورد انتظار برای یافتن پیام دوم که به همان مقدار، hash شود از مرتبهٔ ۲^{n} اجرای Whirlpool می‌باشد.
- تشخیص روابط قاعده داری بین هر ترکیب خطی از بیتهای ورودی و هر ترکیب خطی از نتیجهٔ hash، غیرممکن است و همین‌طور پیش‌بینی اینکه چه بیتهایی از نتیجهٔ hash درصورت تغییر بیتهای مشخصی از ورودی دچار تغییر می‌شوند نیز غیرممکن است (این به معنای استحکام در برابر حملات خطی و تفاضلی می‌باشد).

این ادعاها از آنجایی حاصل می‌شوندکه حاشیهٔ امنیت قابل توجهی نسبت به همهٔ حملات شناخته شده اختیار شده است.

برخلاف همهٔ تحلیل‌ها و بررسی‌ها ممکن است تردیدهایی دربارهٔ وجود راههای نفوذ عمدی جاسازی شده در الگوریتم باقی بماند. NESSIE اعلان طراحان را دراین مورد - مغایر با چیزی که قبلا" بیان شده بود - خواستار شد و بنابراین طراحان Whirlpool اعلام کردند که هیچ ضعف عمدی بوسیلهٔ آنها درPrimitive این الگوریتم وجود ندارد.

◆ دسترسی (Availability)

Whirlpool دارای حق امتیاز ثبت نیست و نخواهد بود و می‌تواند به صورت رایگان برای هر هدفی استفاده شود.

۴-۶-۵-۳. تابع درهم‌سازی RIPEMD-160

◆ خصوصیات کلی

این تابع یک تابع درهم‌سازی ۱۶۰ بیتی می‌باشد که توسط Hans Dobbertin ، Bosselaers Antoon و Bart Preneel طراحی شده است. هدف از طراحی آن این بوده است که به عنوان یک جایگزین امن برای توابع درهم‌سازی ۱۲۸ بیتی

MD4، MD5 و RIPEMD استفاده شود. RIPEMD درقالب پروژهٔ RIPE[1] مربوط به کشورهای اروپایی بوجود آمد. دو دلیل که باعث تفکر درمورد یافتن جایگزین شد از قرار زیرند:

یک hash ۱۲۸ بیتی دیگر امنیت کافی را ارائه نمی‌دهد. یک حملهٔ brute-force جستجوی تصادم روی یک hash ۱۲۸ بیتی به 2^{64} یا $2*10^{19}$ ارزیابی تابع نیاز دارد. در۱۹۹۴، Paul Van Oorschot و Mike Wiener نشان دادند که این حملهٔ brute-force می‌تواند درکمتر از یک ماه با یک هزینهٔ ۱۰ میلیون دلاری انجام شود که انتظار می‌رود این هزینه هر ۱۸ ماه (طبق قانون مور) نصف شود. درسالهای ۹۵ و ۹۶ Hans Dobbertin نیز نقاط ضعفی برای RIPEMD، MD4 و MD5 پیدا کرد.

RIPEMD-160 یک نسخهٔ قوی‌شده RIPEMD با نتیجهٔ hash ۱۶۰ بیتی می‌باشد و انتظار می‌رودکه برای ده سال بعد یا بیشتر امن باشد. فلسفهٔ طراحی آن در استفاده از تجربیات بدست آمده از ارزیابی MD4، MD5 وRIPEMD می‌باشد. این الگوریتم همانند اسلاف‌اش برای پردازنده‌های ۳۲ بیتی بهینه شده است.

- RIPEMD-128 یک جانشین Plug-in برای RIPEMD (یا MD4 و MD5 به دلیل ذکرشده) با حاصل ۱۲۸ بیتی می‌باشد.
- RIPEMD-256 و RIPEMD-320 توسعه‌هایی اختیاری به ترتیب برایRIPEMD-128 وRIPEMD-160 می‌باشند و هدف از آنها بکارگیری درکاربردهایی از توابع درهم‌سازی می‌باشد که نیاز به مقدار hash طولانی‌تری بدون نیاز به سطح امنیت بالاتر دارند.

◆ مقایسهٔ سرعت RIPEMD-160

جدول ۴-۱۵ مقایسه‌ای از کارآیی الگوریتم‌های شبیه MD4 را نشان می‌دهد. فرض شده است که هم کد و هم داده در حافظهٔ نهان روی تراشه قرار دارند. پیاده‌سازیها به اسمبلی 80x86 نوشته شده‌اند و برای پردازندهٔ پنتیوم بهینه شده‌اند.

جدول ۴-۱۵ کارآیی پیاده‌سازی‌های زبان اسمبلی بهینه شدهٔ توابع درهم‌سازی شبیه MD4

الگوریتم	چرخه‌ها	مگابیت برثانیه	مگابایت بر ثانیه	کارایی نسبی
MD4	241	191.2	23.90	1.00
MD5	337	136.7	17.09	0.72
RIPEMD	480	96.0	12.00	0.50
RIPEMD-128	592	77.8	9.73	0.41
SHA-1	837	55.1	6.88	0.29
RIPEMD-160	1013	45.5	5.68	0.24

[1] RACE Integrity Primitives Evaluation

RIPEMD-160 همچنین به همراه RIPEMD-128 و SHA-1 جزء استاندارد بین المللی ISO/IEC[1]، ISO/IEC 10118-3:1998 در مورد توابع درهم‌سازی مختص‌شده[2] می‌باشد.

◆ MAC های بر پایهٔ RIPEMD-160

دو ترکیب از MAC های برپایهٔ توابع درهم‌سازی که هم اینک درون ISO/IEC (ISO/IEC 9797-2) در حال استانداردشدن هستند:

- *MDx-MAC*
- *HMAC-RIPEMD-160*

۴-۵-۶-۴. Tiger : یک تابع درهم‌سازی سریع

◆ خصوصیات کلی

الگوریتم Tiger توسط Ross Anderson از دانشگاه کمبریج و Eli Biham از Technion حیفای اسرائیل ارائه شده است. این الگوریتم طراحی شده‌است که هم سریع و هم امن باشد. هستهٔ آن سه مرحله‌ای می‌باشد که هرکدام از ۸ S-box ۸ به ۶۴ بیتی برای مهیا ساختن یک بهمن[3] غیرخطی قوی به اضافهٔ تعدادی از عملیات رجیستری برای افزایش پخش‌شدگی به هدف استحکام در برابر حملات تفاضلی استفاده می‌کند.

Tiger می‌تواند به طرز مؤثری روی ماشین‌های ۳۲ بیتی و به‌ویژه ۶۴ بیتی پیاده سازی شود. برخلاف SHA-1 می‌تواند از تمام قدرت ماشینهای ۶۴ بیتی هم استفاده کند که حدود ۲/۵ بار سریعتر از SHA-1 می‌باشد. همچنین می‌تواند روی ماشینهای ۱۶ بیتی پیاده‌سازی شود که باز هم انتظار می‌رود که سریعتر از SHA-1 باشد.

خروجی Tiger مقادیر hash ۱۹۲ بیتی می‌باشد. برای تطابق با توابع درهم‌سازی موجود، خروجی آن می‌تواند درصورت لزوم به ۱۶۰ یا ۱۲۸ بیت کوتاه شود. ایجادکنندگان آن معتقدند که حتی این گونه‌های با خروجی کوتاه شده از توابع در حال حاضر با خروجی هم طولشان امن‌تر می‌باشند؛ تصادم برای Tiger/N اساسا" با تلاشی کمتر از O(2N/2) یافت نمی‌شود.

کارآیی این تابع وابسته به موازی‌سازی بالقوه در طراحی آن می‌باشد. درخانواده‌های MD و Snefru هرعمل مستقیماً بستگی به حاصل عمل قبلی دارد بنابراین پردازنده‌های RISC[4] نمی‌توانند به دلیل ازحرکت بازماندن خط لوله[5] بطور مؤثر

[1] International Organization for Standardization /International Electrotechnical Commission
[2] Dedicated
[3] Avalanche
[4] Reduced Instruction Set Computing (or Computer)
[5] Pipeline

مورد استفاده قرارگیرند. در هر مرحلهٔ Tiger، عملیات ۸ جدول lockup می‌توانند به طور موازی انجام شوند و بنابراین کامپایلرها می‌توانند به بهترین وجه از خط لوله استفاده کنند.

مقدار حافظهٔ مورد نیاز برای Tiger تنها کمی بیشتر از اندازهٔ چهار S-box می‌باشد که اگر بتواند در حافظهٔ نهان پردازنده قرار بگیرد، محاسبه حدوداً دوبرابر سریعتر می‌شود (براساس نتایج بدست آمده روی پردازندهٔ Alpha ازشرکت DEC (جدیداً Compaq)).

◆ گونه‌های Tiger

- Tiger/160 : که مقدار hash اولین ۱۶۰ بیت ازحاصل Tiger/192 است و برای تطبیق باSHA و SHA-1 استفاده می شود.
- Tiger/128 : که مقدار hash اولین ۱۲۸ بیت ازحاصل Tiger/192 است و برای تطبیق باMD4، MD5 ، RIPEMD و گونه‌های Snefru و تعدادی از توابع درهم‌سازی مبتنی بر رمزکننده‌های بلوکی می‌باشد.

◆ امنیت Tiger

طراحی اجزاء مختلف آن تقریبا" امنیت مناسبی را ارائه می‌دهند به عنوان مثال:

- غیرخطی بودن که از S-box های ۶۴*۸ بیتی حاصل می‌شود. این ترکیب، خیلی از ترکیب جمع‌ها و XORها بهتر است چراکه در اثر تغییر بیتهایی در ورودی تمام بیتهای خروجی متأثر می‌شوند.
- تمام حملات میانی (در ساختار داخلی) علیه MD*/Snefru به هدف یکی از بلوکهای میانی انجام می‌شوند. افزایش مقدارمیانی به ۱۹۲ بیت به خنثی‌کردن این حملات کمک می‌کند.
- زمانبندی‌کلید تضمین می‌کند که تغییر تعداد کمی از بیتهای پیام بر بیتهای زیادی در خلال گذرهای مختلف تأثیر بگذارد. این عمل کمک می‌کند که در برابر حملات مشابهٔ حملهٔ تفاضلی Dobbertin روی MD4 (که تغییر بیتهای مشخصی در پیام حداکثر دو بیت را در مراحل زیادی متأثر می‌کند) مقاومت کند و ... .

طبق معمول در هنگام ارائهٔ یک Primitive رمزنگاری، ایجادکنندگان اصرار دارند که مردم استحکام و ایمنی Tiger را مطالعه کنند و نتیجه را به آنها اعلام کنند.

۴-۶-۵-۵. خانوادهٔ توابع درهم‌سازی MD (MD2، MD4 و MD5)

این خانواده توسط پروفسور Ron Rivest ازMIT در شرکت RSA Data Security طراحی شده‌اند. MD2 و MD4 نواقص معلوم و مشهوری دارند که استفاده از آنها حتی توسط RSA Data security نیز توصیه نمی‌شود. همچنین وجود

تصادم در تابع فشرده‌سازی MD5 به اثبات رسیده است اما الگوریتم به طورکامل شکسته نشده است. استفاده از MD5 برای کاربردهایی که نیاز به خاصیت عدم تصادم دارند توصیه نمی‌شود اما کاربردهایی که از دیگر خواص آن نظیر یکطرفه بودن و یا خروجی تصادفی استفاده می‌کنند همچنان می‌توانند آن را به کار ببرند. (MD5 در پروتکل‌های زیادی مانند IPSec، PGP، SKIP[1] و SSL استفاده شده است). علیرغم این ضعفها، ایده‌های مورد استفاده در طراحی MD4 ، اساس بسیاری از توابع درهم‌سازی (مانند SHA-1 و RIPEMD) را تشکیل داده‌اند. MD2، MD4 و MD5 به ترتیب در RFC[2] های ۱۳۱۹، ۱۳۲۰ و ۱۳۲۱ تشریح شده‌اند.

۴-۶-۶. خلاصه‌ای از خصوصیات توابع معروف درهم‌سازی

خلاصه‌ای از خصوصیات توابع معروف درهم‌سازی که تاکنون ارائه شده‌اند در جدول ۴-۱۶ آمده است.

جدول ۴-۱۶ خلاصه‌ای از خصوصیات الگوریتم‌های درهم سازی معروف

Name	Ref.	Version	Author(s)	Block Size	Digest Size	Passes	Attack(s)
AR	AR92	1992	?	?	?	?	DK93
Boognish	DGV92a	1992	Daemen	32	up to 160	NA	JD02
Cellhash	DGV91	1991	Daemen, Govaerts, Vandewalle	32	up to 256	NA	?
FFT-Hash I	S91	1991	Schnorr	?	128	4	DBGV91, BGG92
FFT-Hash II	S92	1992	Schnorr	?	128	4	V92
HAVAL	ZPS92	1994	Zheng, Pieprzyk, Seberry	1024	128, 160, 192, 224, 256	3, 4, 5	KP00
MAA	ISO88	1988	ISO	?	?	?	PRO97
MD2	K92	1989	Rivest	512	128	4	RC95
MD4	R90	1990	Rivest	512	128	3	D98
MD5	R92	1992	Rivest	512	128	4	D96
N-Hash	MOI90	1990	Miyaguchi, Ohta, Iwata	128	128	N	BS91
Panama	DC98	1998	Daemen, Clapp	256	unlimited	NA	RRPV01
RIPEMD	RIPE92	1990	Preneel	?	128	4	D97

[1] Simple Key-management for Internet Protocol
[2] Request For Comments

RIPEMD-128	DBP96	1996	Dobbertin, Bosselaers, Preneel	?	128	4	?
RIPEMD-160	DBP96	1996	Dobbertin, Bosselaers, Preneel	?	160	5	?
SHA-0	NN91	1991	NIST/NSA	512	160	4	CJ98
SHA-1	NN02	1993	NIST/NSA	512	160	4	?
SHA-256	NN02	2000	NIST/NSA	512	256	64(*)	?
SHA-384	NN02	2000	NIST/NSA	1024	384	80(*)	?
SHA-512	NN02	2000	NIST/NSA	1024	512	80(*)	?
Snefru	M90	1990	Merkle	512 - *m*	*m* = 128, 256	?	BS93
StepRightUp	JD95	1995	Daemen	256	256	NA	RRPV01
Subhash	DGV92b	1992	Daemen	32	up to 256	NA	?
Tiger	AB96	1996	Anderson, Biham	512	192	3	?
WHIRLPOOL	BR00	2000	Barreto, Rijmen	512	512	10(*)	?
Name	**Ref.**	**Version**	**Author(s)**	**Block Size**	**Digest Size**	**Passes**	**Attack(s)**

(*) SHA-256, SHA-384, SHA-512, and WHIRLPOOL are partitioned in rounds rather than passes.

References for the Previous Table

[AB96]

R. Anderson and E. Biham, "Tiger: A Fast New Hash Function", *Fast Software Encryption - FSE'96*, LNCS 1039, Springer-Verlag (1996), pp. 89--97.

[AR92]

ISO N179, "AR Fingerprint Function", working document, ISO-IEC/JTC1/SC27/WG2, International Organization for Standardization, 1992.

[BGG92]

T. Baritaud, H. Gilbert, and M. Girault, "F.F.T. hashing is not collision-free." *Advances in Cryptology - Eurocrypt'92*, LNCS 658, Springer-Verlag (1992), pp. 35--44.

[BR00]

P. S. L. M. Barreto and V. Rijmen, "The Whirlpool Hashing Function", First open NESSIE Workshop, Leuven, Belgium, 13--14 November 2000.

[BRS02]

J. Black, P. Rogaway, and T. Shrimpton, "Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV", *Advances in Cryptology - CRYPTO'2002*, LNCS 2442, Springer-Verlag (2002), pp. 320--335.

[BS91]

E. Biham and A. Shamir, "Differential cryptanalysis of Feal and N-Hash", *Advances in Cryptology - Eurocrypt'91*, LNCS 547, Springer-Verlag (1991), pp. 1-16.

[BS93]

E. Biham and A. Shamir, "Differential Cryptanalysis of the Data Encryption Standard", Springer-Verlag (1993).

[CJ98]

F. Chabaud and A. Joux, "Differential Collisions in SHA-0", *Advances in Cryptology - Crypto'98*, LNCS 1462, Springer-Verlag (1998), pp. 56--71.

[D96]

H. Dobbertin, "The Status of MD5 After a Recent Attack", *CryptoBytes* 2:2 (1996), pp. 1--6.

[D97]

H. Dobbertin, "RIPEMD with Two-Round Compress Function is Not Collision-Free", *Journal of Cryptology* 10:1 (1997), pp. 51--70.

[D98]

H. Dobbertin, "Cryptanalysis of MD4", *Journal of Cryptology* 11:4 (1998), pp. 253--271.

[DBGV91]

J. Daemen, A. Bosselaers, R. Govaerts, and J. Vandewalle, "Collisions for Schnorr's Hash Function FFT-Hash", *Advances in Cryptology - Asiacrypt'91*, LNCS 739, Springer-Verlag (1993), pp. 447--480.

[DBP96]

H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160, a strengthened version of RIPEMD", *Fast Software Encryption - FSE'96*, LNCS 1039, Springer-Verlag (1996), pp. 71--82.

[DC98]

J. Daemen and C. Clapp, "Fast Hashing and Stream Encryption with Panama", *Fast Software Encryption - FSE'98*, LNCS 1372, Springer-Verlag (1998), pp. 60--74.

[DGV91]

J. Daemen, R. Govaerts, and J. Vandewalle, "A Framework for the Design of One-Way Hash Functions Including Cryptanalysis of Damgård's One-Way Function Based on Cellular Automata", *Advances in Cryptology - Asiacrypt'91*, LNCS 739, Springer-Verlag (1993), pp. 82--96.

[DGV92a]

J. Daemen, R. Govaerts, and J. Vandewalle, "Fast Hashing Both in Hard- and Software", *ESAT-COSIC Report* 92-2, Department of Electrical Engineering, Katholieke Universiteit Leuven, April 1992.

[DGV92b]

J. Daemen, R. Govaerts, and J. Vandewalle, "A Hardware Design Model for Cryptographic Algorithms", *European Symposium on Research in Computer Security - ESORICS*, 1992, pp. 419--434.

[DK93]

I. B. Damgård, and L. R. Knudsen, "The breaking of the AR Hash Function", *Advances in Cryptology - EUROCRYPT'93*, LNCS 765, Springer Verlag (1994), pp. 286--292.

[ISO88]

ISO Standard 8731-2, 1988. More information can be found here.

[JD95]

J. Daemen, Doctoral dissertation, Katholiek Universiteit Leuven, 1995.

[JD02]

J. Daemen, personal communication, 2002 (if you are curious, it merely states that Boognish is "certainly weak").

[K92]

B. Kaliski, "The MD2 Message-Digest Algorithm", RFC 1319 (1992).

[KP00]

P. Kasselman and W. Penzhorn, "Cryptanalysis of Reduced Version of HAVAL", Electronics Letters, Vol. 36, No. 1, January 2000, pp. 30--31.

[M90]

R. C. Merkle, "A Fast Software One-Way Hash Function", *Journal of Cryptology* 3:1 (1990), pp 43--58.

[MOI90]

S. Miyaguchi, K. Ohta, and M. Iwata, "128-bit hash function (N-hash)", *NTT Review*, vol. 2 (no. 6), Nov. 1990, pp. 128--132.

[NN91]

NIST/NSA, "FIPS 180" (superseded by FIPS 180-1 and FIPS 180-2). See also NIST's Secure Hashing site.

[NN02]

NIST/NSA, "FIPS 180-2: Secure Hash Standard (SHS)", August 2002. See also NIST's Secure Hashing site.

[PRO97]

B. Preneel, V. Rijmen, and P. van Oorschot, "Security analysis of the Message-Authenticator Algorithm (MAA)", *European Transactions on Telecommunications*, Vol. 8, No. 5 (Sept./Oct. 1997), pp. 455--470.

[R90]

R. L. Rivest, "The MD4 Message-Digest Algorithm", *Advances in Cryptology - Crypto'90*, LNCS 537, Springer-Verlag (1990), pp. 303--311.

[R92]

R. L. Rivest, "The MD5 Message-Digest Algorithm", RFC 1321 (1992).

[RC95]

N. Rogier and P. Chauvaud, "The compression function of MD2 are not collision-free", workshop record, 2nd Workshop on Selected Areas in Cryptography (SAC'95), Ottawa, Canada, May 1819, 1995.

[RIPE92]

Research and Development in Advanced Communication Technologies in Europe, "RIPE Integrity Primitives: Final Report of RACE Integrity Primitives Evaluation (R1040)", RACE, June 1992.

[RRPV01]

V. Rijmen, B. Van Rompay, B. Preneel, and J. Vandewalle, "Producing Collisions for PANAMA", *Fast Software Encryption - FSE'2001*, LNCS 2355, Springer-Verlag (2002), pp. 37--51.

[S91]

C. Schnorr, "FFT-Hash, An Efficient Cryptographic Hash Function", *Crypto'91 rump session*, unpublished manuscript, 1991.

[S92]

C. Schnorr, "FFT-Hash II, efficient cryptographic hashing", *Advances in Cryptology - Eurocrypt'92*, LNCS 658, Springer-Verlag (1992), pp. 45--54.

[V92]

S. Vaudenay, "FFT-Hash-II is not yet Collision-free", *Advances in Cryptology - Crypto'92*, LNCS 740, Springer (1993), pp. 587--593.

[ZPS92]

Y. Zheng, J. Pieprzyk, and J. Seberry, "HAVAL - a one-way hashing algorithm with a variable length of output", *Advances in Cryptology - Auscrypt'92*, LNCS 718, Springer-Verlag (1993), pp. 83--104.

۴-۷. الگوریتم‌های رمزنگاری کلیدعمومی (نامتقارن)

گفته می‌شود که "رمزنگاری کلید عمومی" (PKC)[1] ، مهمترین پیشرفت رمزنگاری در۳۰۰-۴۰۰ سال اخیر بوده است. رمزنگاری کلید عمومی درعصر دیجیتال امروزه در همه جا حاضر است. بعضی روشها و الگوریتم‌های کلید عمومی نظیر PGP وRSA در صنعت کامپیوتر معروف و مشهورند و به منزلهٔ ستونی در مهیا کردن امنیت داده و محافظت از جامعیت داده به‌شمار می‌روند. بدون آن، ما شدیداً در مبادلهٔ اطلاعات حساس و محرمانهٔ بین دو طرف یا بیشتر دچار مشکل می‌شویم. درنیمهٔ آخر قرن بیستم بخصوص با رشد محبوبیت اینترنت، رمزنگاری یک نقش برتر را ایفا می‌کند. در بخش تجارت، رمزنگاری کلید عمومی به امری واجب تبدیل شده است. از آنجاییکه انواع دزدی‌ها، جعل سند و hack تا حد زیادی رایج شده‌اند، باید هر نوع اقدام پیشگیرانه‌ای را انجام داد. تقریباً تمام معاملات تجاری روی‌خط (On-Line) به‌شدت به مهندسی رمزکردن و رمزگشایی پیامها وابسته است.

گزافه‌گویی نیست اگر ادعاکنیم که رمزنگاری کلید عمومی عاملی است که رمزنگاری را از چیزی که تنها بوسیلهٔ مردمی استفاده می‌شد که نیاز قوی برای محرمانگی یا علاقه و تمایل به خود رمزنگاری داشتند، به چیزی که روزمره بوسیلهٔ افراد زیادی برای اهداف عملی نظیر انجام خرید با کارتهای اعتباری از طریق اینترنت استفاده می‌شود؛ تبدیل نمود.

اساس رمزنگاری کلیدعمومی ساده است. برای مثال، یک کاربر یک جعبهٔ قفل‌دار و دو مجموعه‌کلید دارد. یک کلید، کلید عمومی است که می‌تواند جعبه را قفل کند اما نمی‌تواند آن را بازکند. کلید دیگر یک کلید خصوصی است که می‌تواند جعبه را باز کند اما نمی‌تواند آن را قفل کند. دوکلید، متفاوت هستند و بنابراین یک کلید با استفاده از دیگری نمی‌تواند تولید شود. دوست شما که می‌خواهد به شما یک پیغام بفرستد، جعبه قفل‌دار را می‌گیرد، پیام را داخل آن قرار می‌دهد و با کلید عمومی آن را قفل می‌کند. بعد از رسیدن جعبه به دست شما، شما برای بازکردن آن از کلید خصوصی‌تان استفاده می‌کنید. درحقیقت، هرکسی که می‌خواهد به شما یک پیام را بفرستد، کلید عمومی شما را گرفته، -که شما آن را به آزادی پخش کرده‌اید- و پیامهایش را در جعبهٔ قفل‌دار، قفل می‌کند و به شما می‌فرستد. شما کلید خصوصی را به طور امن نگه می‌دارید و هرجعبه را برای رمزگشایی پیامها باز می‌کنید.

حیلهٔ[2] به‌کار گرفته شده در PKC به وجود به اصطلاح توابع یکطرفه بستگی دارد یا توابع ریاضی که برای محاسبه ساده هستند درحالیکه معکوسشان برای محاسبه مشکل است. مثلاً از موارد مورد استفاده در PKC ، سادگی ضرب و توان‌رساندن در مقابل سختی نسبی تجزیه و لگاریتم‌ها می‌باشد. از آنجائیکه دوکلید لازم است به آن رمزنگاری نامتقارن گفته می‌شود.

الگوریتم‌ها و قراردادهای رمزنگاری به سختی درست می‌شوند، بنابراین هیچ وقت نباید از ایده‌های شخصی برای آن استفاده نمود. در عوض، هرجایی که امکان دارد باید از الگوریتم‌ها وقراردادهایی که بطور گسترده استفاده می‌شوند، استفاده کرد.

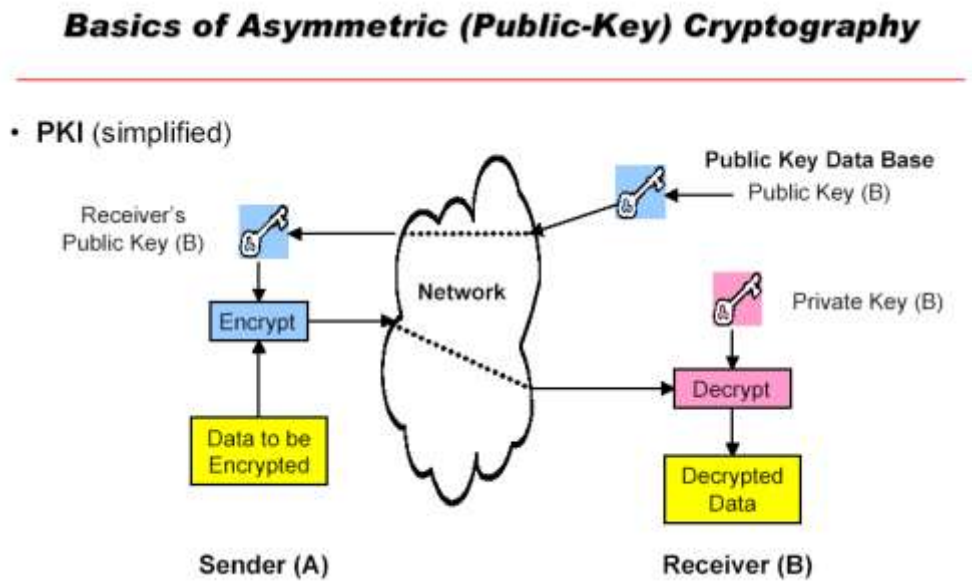
[1] Public Key Cryptography
[2] Trick

چراکه به شدت بررسی شده‌اند و پذیرفته شده‌اند که امن می‌باشند. مشخصاً، الگوریتم رمزنگاری را نباید خود طراحی کرد مگر اینکه درعلم رمزنگاری و ریاضیات وابسته خبره باشیم. درست‌کردن الگوریتم‌های رمزی که نسبتاً خوب است وظیفهٔ خبرگان می‌باشد.

تعدادی از الگوریتم‌ها دارای حق امتیاز هستند. هرچندکه صاحبان اجازهٔ "استفادهٔ آزاد" را اکنون می‌دهند، - بدون هیچ قرارداد امضاء شده‌ای- آنها همیشه می‌توانند بعداً فکرشان را عوض کنند و شما را در خطر بزرگی قرار دهند. بطورکلی از تمام الگوریتم‌های دارای حق‌امتیاز باید اجتناب شود. در تقریباً همهٔ زمینه‌ها موارد بدون حق امتیاز وجود دارند که از لحاظ تکنیکی به‌خوبی موارد دارای امتیاز و چه بسا بهتر هستند. با انجام چنین عملی از تعداد زیادی از مشکلات قانونی اجتناب خواهد شد.

۴-۷-۱. پیدایش رمزنگاری کلیدعمومی

"سیستم‌های رمزنگاری"[1] کلید عمومی در اواخر دهۀ ۱۹۷۰ با یاری‌گرفتن از نظریۀ پیچیدگی[2] حوالی همان زمان اختراع شدند. در آن زمان ملاحظه شده بود که بر پایۀ مسأله‌ای که چندین سال برای حل آن زمان نیاز است می‌توان یک سیستم رمزنگاری ایجاد کرد که دو کلید داشته باشد: یک کلید محرمانه و یک کلید عمومی. از کلید عمومی برای رمزکردن پیامها و از کلید خصوصی برای رمزگشایی استفاده شود. بنابراین صاحب کلید خصوصی تنها فردی خواهد بود که قادر به رمزگشایی پیامها می‌باشد، اما هرکس با داشتن کلید عمومی می‌تواند پیامها را درمحرمانگی بفرستد. اساس رمزنگاری نامتقارن (کلید عمومی) درشکل ۴-۲۹ دیده می‌شود (امروزه غالباً این نوع رمزنگاری در ساختاری به نام PKI[3] مورد استفاده قرار می‌گیرد).



شکل ۴-۲۹ اساس رمزنگاری کلیدعمومی (نامتقارن)

ایدۀ دیگر که مورد نظر قرارگرفت در مورد مبادلۀ کلید بود. در یک ارتباط دوطرفه مفید خواهد بود که یک کلید محرمانۀ مشترک برای "رمزکردن توده"[4] برای استفاده از یک سیستم رمزنگاری کلید متقارن تولید شود.

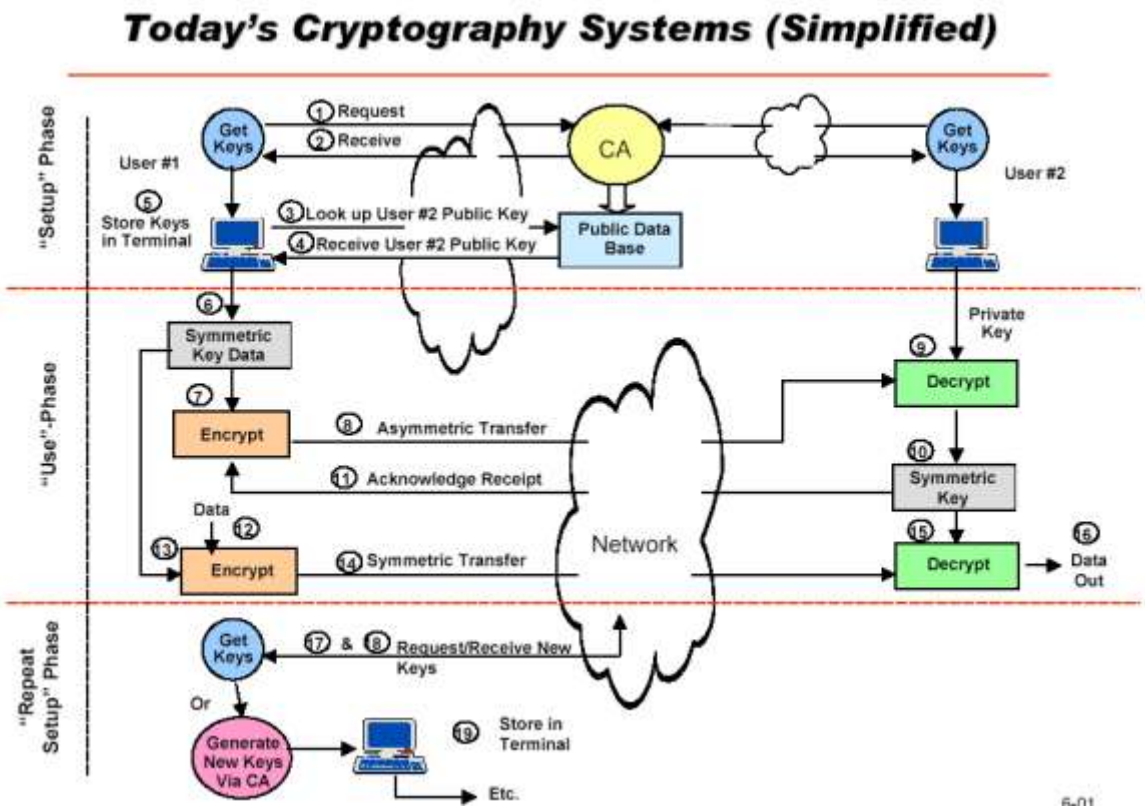
درسیستم‌های کلید متقارن به $N(N+1)/2$ کلید برای N طرف ارتباط نیاز است که با افزایش تعداد افراد، تعداد کلیدهای مورد نیاز به سرعت افزایش می‌یابد و مسألۀ مدیریت کلید در این‌گونه سیستم‌ها امری بسیار پیچیده خواهد بود. ازطرفی اگر دشمن قادر به کشف کلید ارتباطی بین دو ایستگاه باشد، علاوه بر آنکه شبکۀ ارتباطی امنیت خود را از دست می‌دهد، دشمن خواهد توانست اطلاعات و پیامهای جعلی را درشبکه تزریق کند زیرا در سیستم‌های رمزنگاری متقارن امکان تشخیص فرستندۀ پیام وجود ندارد و مبدأ ارسال اطلاعات با فرض امنیت کلید، فرستندۀ مورد اطمینان هر ایستگاه می‌باشد.

[1] Cryptosystems
[2] Complexity
[3] Public Key Infrastructure
[4] Bulk Encryption

به‌دلیل مشکلات بالا دریک شبکهٔ ارتباطی، همواره این سؤال در ذهن طراحان سیستم‌های رمزنگاری وجود داشته است که آیا می‌توان سیستم‌های رمزنگاری را چنان طراحی کرد که مشکل مخفی ماندن کلید رمز را در شبکه حل کند؟ یا اینکه آیا می‌توان سیستم رمزنگاری طراحی کرد که دشمن حتی درصورت آگاهی از کلید رمز ارسال پیامها، نتواند به محتوای اصلی پیام دست یابد؟

نخستین پاسخ‌ها به این سؤالات را Martin Hellman استاد دانشگاه استنفورد و Whitfield Diffie در سال ۱۹۷۶ ارائه کردند. آنها در مقالهٔ خود با عنوان "نگرشهای جدید در رمزنگاری"[1] منتشرشده در نوامبر ۱۹۷۶ (نشریهٔ IEEE Transaction on Information Theory)، این نظریه را بیان کردند که می‌توان کانال امن برای ارسال را از سیستم رمزنگاری خارج کرد و یا به عبارت دیگر کلید رمزکردن را در اختیار عموم گذارد. در واقع Diffie و Hellman از ایده‌هایی از نظریهٔ اعداد برای ایجاد یک قرارداد مبادلهٔ کلید استفاده کردند. با این اندیشه، تحول شگرفی در رمزنگاری با عنوان "سیستم‌های رمزنگاری با کلید عمومی" ایجاد و نظریهٔ آن بسط داده شد. البته امروزه اعتقاد بر این است که یک تیم در "مراکز ارتباطات دولتی"(GCHQ)[2] کشور انگلستان برای اولین بار PKC را در اوایل دههٔ ۷۰ ایجاد کردند اما بخاطر طبیعت کار، GCHQ یادداشتهای اصلی را سری نگاهداشت. اولین سیستم رمزنگاری عملی بر این پایه را در سال ۱۹۷۸، Rivest، Shamir و Adleman به نام RSA (که اولین سیستم رمزنگاری کلید عمومی واقعی با توانایی رمزکردن و امضاهای دیجیتالی بود) طراحی و به نظرخواهی نهادند. بعداً چندین سیستم رمزنگاری عمومی با استفاده از ایده‌های زیرساختاری متفاوت زیادی (نظیر"مسألهٔ کوله‌پشتی"[3]، گروههای متفاوت روی میدانهای متناهی و شبکه‌ها[4]) ارائه شدند. طولی نکشید که ثابت شد تعداد زیادی از آنها ناامن هستند. به هرحال قرارداد Diffie-Hellman و الگوریتم RSA به نظر می‌رسد که به عنوان دو الگوریتم قوی تا کنون باقی مانده‌اند. شکل ۴-۳۰، ساده‌شدهٔ مراحل ساده شدهٔ اعمال انجام شده در سیستم‌های رمزنگاری امروزه را نشان می‌دهد.

[1] New Directions in Cryptography
[2] Government Communications Head Quarters
[3] Knapsack problem
[4] Lattices



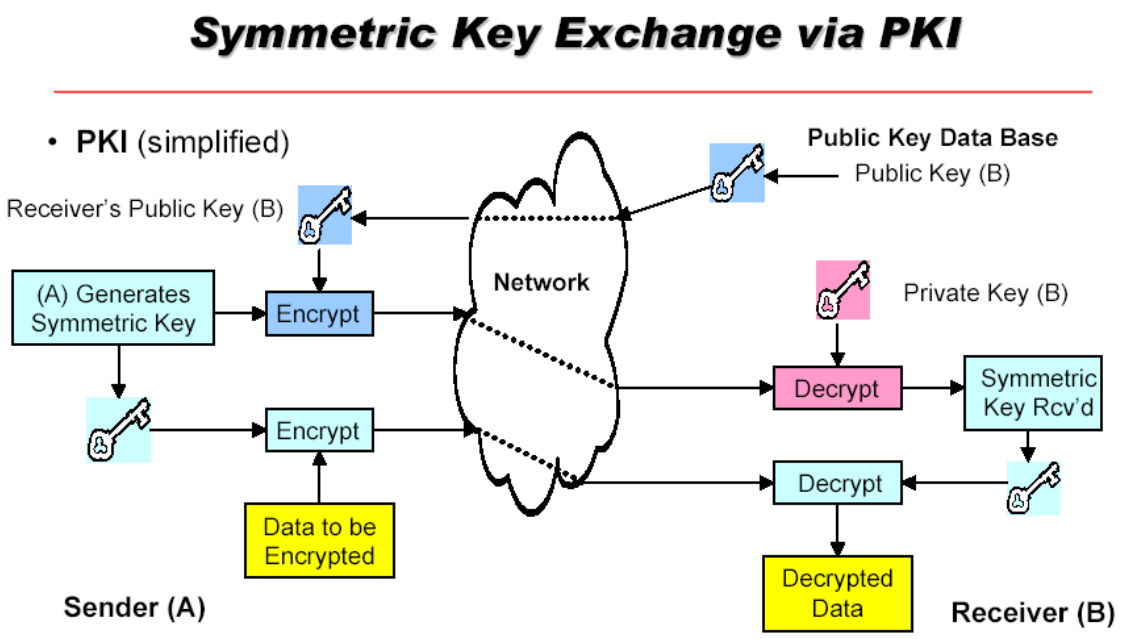
شکل ۴-۳۰ سیستم‌های رمزنگاری امروزه

هدف اصلی رمزکردن کلید عمومی مهیا ساختن پوشیدگی[1] یا محرمانگی[2] می‌باشد. از آنجاییکه عمل رمزکردن یک دانش عمومی است (هر کس با داشتن کلید عمومی قادر به انجام آن است)، رمزکردن کلید عمومی به تنهایی اعتبارسنجی منبع داده یا جامعیت داده را مهیا نمی‌کند. چنین اطمینان‌هایی باید از طریق استفاده از تکنیک‌های اضافی شامل کدهای اعتبارسنجی پیام و امضاهای دیجیتالی بدست آیند.

موارد استفاده از رمزنگاری کلید عمومی (بسته به نوع کاربرد) در زیر لیست شده‌اند:

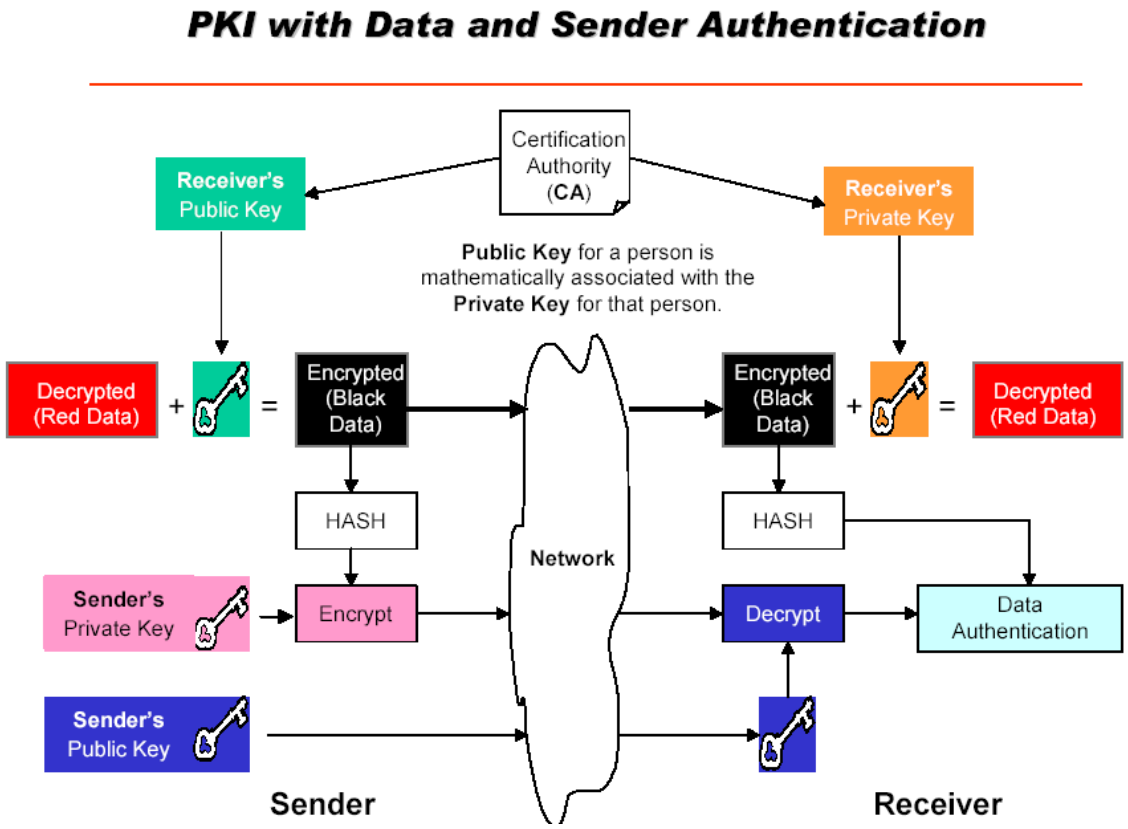
- مبادلهٔ کلید (فائق‌آمدن بر مشکلات ناشی از مدیریت کلید در سیستمهای رمزنگاری متقارن). شکل ۴-۳۱ نشان‌دهندهٔ این کاربرد رمزنگاری کلید عمومی می‌باشد.

[1] Privacy
[2] Confidentiality



شکل ۴-۳۱ مبادلهٔ کلید متقارن از طریق PKI

- امضای دیجیتالی. این کاربرد برای اعتبارسنجی داده و فرستنده استفاده می‌شود. در دنیای دیجیتال امروزه اعتبارسنجی به مهمی پوشیدگی داده می‌باشد. در حالات زیادی، رمزکردن داده در صورتیکه طرف دیگر نتواند اعتبارسنجی شود، بدون معنی است. بنابراین، اعتبارسنجی قوی، یک امر ملزم می‌باشد. شکل ۴-۳۲ چگونگی استفاده از رمزنگاری کلید عمومی به هدف اعتبارسنجی داده و فرستنده را نشان می‌دهد.



شکل ۴-۳۲ کاربرد PKC در اعتبارسنجی داده و فرستنده

- پوشیدگی (رمزکردن). از این کاربرد تنها در رمزکردن حجمهای کوچک داده استفاده می‌شود.

بر این اساس الگوریتم‌های کلید عمومی را دسته‌بندی می کنند که در ادامه به آن پرداخته شده است.

۴-۷-۲. دسته‌بندی الگوریتم‌های کلید عمومی از لحاظ کاربرد

- *PKDS*[1]
  - مورد استفاده در انتقال یک قطعهٔ اطلاعات به صورت امن
  - مقدار به‌طور معمول بعنوان یک کلید جلسه برای یک روش کلید خصوصی استفاده می‌شود.
- *PKE*[2]
  - مورد استفاده برای رمزکردن هر طول پیام
  - هرکسی می‌تواند از کلید عمومی برای رمز کردن پیامها استفاده کند.
  - صاحب کلید از کلید خصوصی برای رمزگشایی پیام استفاده می کند.

[1] Public Key Distribution Schemes
[2] Public Key Encryption

- هر روش رمز کلید عمومی با استفاده از کلید جلسه به جای پیام می‌تواند به عنوان یک PKDS استفاده شود.

- *روشهای امضاء*
  - استفاده برای تولید یک امضای دیجیتالی برای یک پیام
  - صاحب کلید، ازیک کلید خصوصی برای امضاء استفاده می کند.
  - هرکسی می‌تواند کلید عمومی را برای بررسی اعتبار امضاء استفاده کند.

در واقع در این حالت، تمییز دادن بین دسته‌های مختلف الگوریتم‌های کلید عمومی، برحسب استفادهٔ آنها صورت می گیرد.

۴-۷-۳. مقایسهٔ رمزنگاری کلید عمومی با رمزنگاری کلید متقارن

◆ مزایا

۱. به کانال مورد اطمینان (امن) نیاز ندارند.

۲. طولهای کلید متغیر می‌پذیرند.

۳. بسته به مورد استفاده، یک جفت کلید عمومی/خصوصی ممکن است برای یک دورهٔ زمانی قابل توجه، قابل استفاده باشد.

۴. غالب روشهای کلید عمومی در مکانیزمهای امضای دیجیتالی کارآ استفاده می‌شوند. کلید مورد استفاده برای تأیید، نوعاً خیلی کوچکتر از آنچه که برای کلید متقارن می‌باشد، است.

۵. تنها کلید خصوصی باید محرمانه بماند (اعتبار و صحت کلیدهای عمومی به هرحال باید تضمین شود).

۶. تعداد کلیدهایی که توسط هرکاربر باید مدیریت شود خیلی کمتر است.

◆ معایب

۱. عمل رمزکردن، به علت استفاده از ریاضیات پیچیده، کند است؛ به‌همین دلیل، اساساً برای نرخ دادهٔ کم استفاده می‌شود.

۲. ciphertext خیلی بزرگتر از plaintext می‌باشد.

۳. اندازهٔکلیدها نوعاً خیلی بزرگتر است و همچنین، اندازهٔ امضاهای کلید عمومی بزرگتر از برچسب[1]های مورد استفاده در کلید متقارن برای اعتبارسنجی منبع داده می‌باشد.

۴. به سادگی در سخت افزار پیاده‌سازی نمی‌شوند.

[1] Tag

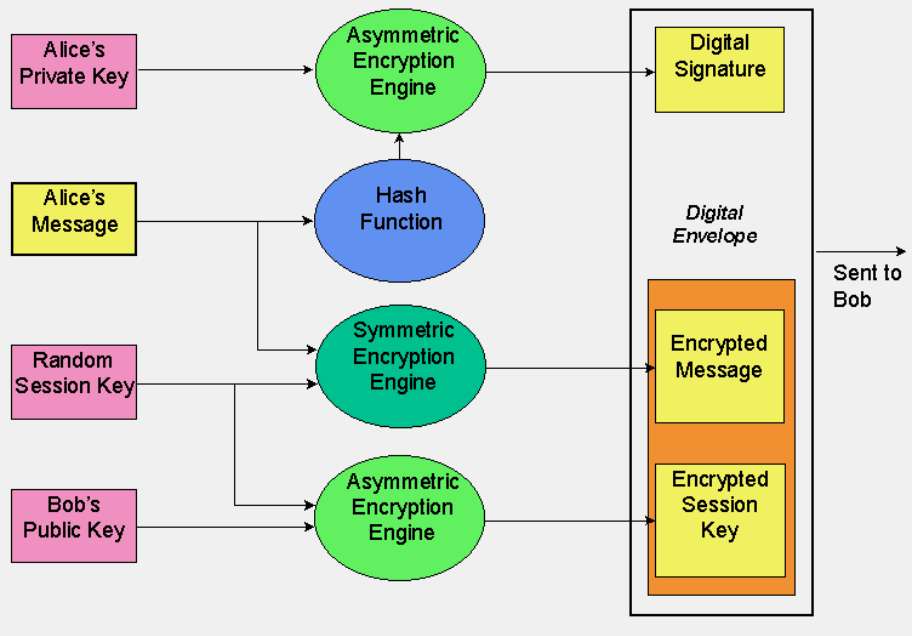
۵. اعتبارسنجی کلیدهای عمومی مورد نیاز است.

۶. "تدارکات مدیریت کلید"[1] نیاز دارد.

۷. هیچ روش کلید عمومی که امنیت آن کاملا ثابت شده باشد، وجود ندارد (نظیر چیزی که در مورد رمزکننده‌های بلوکی نیز گفته شده‌است).

با توجه به مزایا و معایب بیان‌شده، باید از ترکیب این دو دستهٔ رمزنگاری استفاده کرد تا از مزایای امنیت کلید عمومی و مزایای سرعت کلید متقارن استفاده شود.

شکل ۴-۳۳ هر سه دستهٔ الگوریتم‌های رمزنگاری را در کنار یکدیگر قرار داده و نشان می‌دهد که چگونه یک روش رمزنگاری، ترکیبی از همهٔ الگوریتم‌های رمزنگاری را برای ایجاد ارتباط امن شامل امضای دیجیتالی و پاکت دیجیتالی به‌کار می‌برد. در این مثال، Alice فرستندهٔ پیام و Bob گیرنده می‌باشد.



شکل ۴- ۳۳ یک کاربرد نوعی از سه روش رمزنگاری برای ارتباط امن

۴-۷-۴. مثالهایی از کاربردهای روزمرهٔ الگوریتم‌های کلید عمومی

- *ATM*[2] ها

اولین مثالی که از سیستم‌های رمزنگاری مورد استفادهٔ وسیع به ذهن می‌رسد، ATM ها (یا ABM[3]ها) می‌باشد. در این کاربرد، سیستم‌های رمزنگاری خدماتی را هم به مشتری و هم به بانک صادرکنندهٔ کارت ارائه می‌دهند. اولاً، غیرعملی است

[1] Key administration logistics
[2] Automatic Teller Machine
[3] Automatic Banking Machine

که هرعدد PIN[1] کاربر در هر ATM ذخیره گردد. بنابراین PIN داده شده به ATM باید از طریق یک شبکه به دفتر مرکزی بانک برای بررسی منتقل شود. محرمانگی این PIN مشخصاً در زمانی که منتقل می‌شود، امری ضروری است. ثانیاً، وقتی یک مقدار پول درخواست می‌شود، جامعیت این اطلاعات باید حفظ شود. در غیر اینصورت مقدار تقاضا شده می‌تواند تغییرکند و پول از حسابها به صورت غیرقانونی برداشت شود.

- *کارتهای تلفن*

استفاده از کارتهای تلفن با "مقدار ذخیره شده" تا حد زیادی گسترده است. در این کاربرد، رمزشدهٔ مقدار پول باقیمانده روی کارت به شکل الکترونیکی روی کارت ذخیره می‌گردد. فرض کنید که جامعیت این اطلاعات محافظت نشده بود، در این صورت هرکس می‌توانست اطلاعات ذخیره شده را تغییر دهد و بنابراین مقدار ذخیره شده روی کارت تازه شود که نتیجهٔ آن دسترسی مجانی به شبکهٔ تلفن است.

در این مورد جامعیت داده بوسیلهٔ یک سیستم رمزنگاری مهیا می‌شود تا اپراتور شبکهٔ تلفن را از گول‌خوردن محافظت کند.

- *شبکه‌های تلفن سلولی*

محیط دیگری از ارتباطات تلفنی که آسیب‌پذیر است، استفاده از تلفنهای سلولی می‌باشد. این موضوع اکنون تنها بوسیلهٔ "مهیاکنندگان خدمات"[2] نظارت می‌شود. نسل بعدی تلفنهای سلولی از سیستم‌های رمزنگاری برای جلوگیری از سوء استفاده‌ها، استفاده خواهند کرد. اتحادیهٔ صنعت ارتباطات راه دور سلولی تخمین می‌زند که تلفنهای سلولی نسل فعلی بدون امنیت، استفادهٔ غیر قانونی به هزینهٔ بین ۱/۵ تا ۲/۵میلیون دلار را در هرسال به همراه دارد.

- *دسترسی "راه دور"[3] به سیستم*

یک تمایل رو به ازدیاد در تجارت، استفادهٔ راه دور از تسهیلات محاسباتی می‌باشد، کارمندان زیادی در دفترکار خانه یا سیار بوسیلهٔ کامپیوتر با اتصال راه دور، به انجام کارهای خود می‌پردازند وبسیاری موارد دیگر.

- *کارتهای هوشمند*[4]

یک کاربرد جالب که هم اینک درحال گسترده‌شدن می‌باشد، استفاده از کارتهای هوشمند (همچنین به نام chipcardها) می‌باشد. از لحاظ اندازه، به همان اندازهٔ کارتهای اعتباری استاندارد است، اما یک پردازندهٔ کوچک دارند. با پیشرفت فن‌آوری کلید عمومی، هم اینک نصب یک سیستم کلید عمومی روی این کارتها با هزینهٔ کم امکان‌پذیر است.

[1] Personal Identification Number
[2] Service Providers
[3] Remote
[4] Smart cards

۴-۷-۵. "مجموعهٔ اصطلاحات فنی"[1]

جزء اساسی یک سیستم رمزنگاری کلید عمومی یک مسألهٔ "ازنظر محاسباتی مشکل" می‌باشد. امنیت این نوع سیستم‌های رمزنگاری بر پایهٔ این حقیقت است که کلید خصوصی تنها -با حل این مسأله مشکل- از کلید عمومی قابل محاسبه است. اکنون با چند اصطلاح مورد استفاده در رمزنگاری کلید عمومی آشنا می‌شویم:

- الگوریتم. یک الگوریتم یک توصیف صریح از چگونگی انجام یک محاسبهٔ مشخص یا حل مسأله می‌باشد. کارآیی یک الگوریتم می‌تواند با تعداد گامهای مقدماتی که طول می‌کشد تا مسأله حل گردد، اندازه‌گیری شود. بنابراین اگر ادعا شود که انجام الگوریتم زمانی از مرتبهٔ$O(n)$ طول می‌کشد منظور این است که به اندازهٔ n گام(قدم) مقدماتی زمان طول می‌کشد اما مشخص نمی‌شود که یک گام چقدر طول می‌کشد.

- پیچیدگی محاسباتی. یک مسأله دارای زمان چند جمله‌ای یا P-time است اگر بتواند بوسیلهٔ یک الگوریتم که کمتر از $O(n^t)$ گام نیاز دارد، حل شود. t یک عدد متناهی و متغیرn اندازهٔ مسأله می‌باشد. اگر برای یک مسأله بتوان یک راه حل در زمان چند جمله‌ای یافت آنگاه گفته می‌شود که این مسأله NP[2] می‌باشد. مجموعهٔ مسائلی که در NP قرار می‌گیرند خیلی بزرگ است و شامل "تجزیهٔ عدد صحیح"[3] است.

یک مسأله، NP-hard است اگر مسألهٔ دیگری شبیه به آن وجود نداشته باشد که دارای راه حل NP باشد. الگوریتمی با زمان چندجمله‌ای شناخته شده برای هیچ‌کدام از مسائل NP-hard موجود نیست و اعتقاد بر این است که چنین الگوریتم‌هایی درحقیقت موجود نیستند.

در رمزنگاری کلید عمومی، حمله‌کننده تمایل به حل نمونه‌های مشخصی ازیک مسأله (مثلاً تجزیهٔ یک عدد داده شده)، به جای تهیهٔ یک راه‌حل عمومی (یک الگوریتم برای تجزیهٔ هرعدد ممکن به طورمؤثر) دارد. این مطلب کمی نگرانی برای خبره‌های رمزنگاری ایجاد می‌کند، چراکه بعضی نمونه‌های یک مسأله که در حالت کلی NP-hard می‌باشد ممکن است به‌سادگی قابل حل باشد.

- اعداد اول. یک عدد اول، عددی است که هیچ مقسوم علیهی غیر از خودش و ۱ نداشته باشد. بنابراین اعداد صحیح ۲، ۳، ۵، ۷، ۱۱ و... اول هستند. تعداد اعداد اول نامتناهی است و (یکی از) بزرگترین اعداد اول شناخته شده (۱- $۲^{۶۹۷۲۵۹۳}$) می‌باشد.

[1] Terminology
[2] Non-deterministic Polynomial-time
[3] Integer factorization

- تجزیه[1](عامل‌گیری). هرعدد صحیح می‌تواند به صورت منحصر بفردی از ضرب اعداد اول نشان داده شود. بعنوان مثال ۵ * ۲ = ۱۰ و منحصر بفرد است (بجز ترتیب عاملهای۲ و ۵). هنر تجزیه تقریباً به قدمت خود ریاضیات می‌باشد؛ هرچندکه، مطالعهٔ الگوریتم‌های سریع برای تجزیه تنها چند دهه قدمت دارد.

یک الگوریتم ممکن برای تجزیهٔ یک عدد صحیح، تقسیم آن برهمهٔ اعداد اول کوچکتر به صورت تکراری تا زمانیکه عدد باقیمانده اول باشد، می‌باشد. این الگوریتم تنها برای اعداد صحیحی که از مرتبهٔ کوچکتر از 10^{16} می‌باشند؛ کاراست. برای چنین اعدادی نیاز به بررسی همهٔ اعداد اول تا 10^{8} می‌باشد. در سیستم‌های رمزنگاری کلید عمومی بر پایهٔ مسألهٔ تجزیه، اعداد از اندازهٔ 10^{300}می‌باشند که نیاز به بررسی همهٔ اعداد اول تا 10^{150} دارد. حدود 10^{147} تا از چنین اعداد اولی بر طبق نظریهٔ اعداد اول موجودند. این تعداد به مراتب از تعداد اتمهای موجود در جهان بزرگتر است و با هر تقلا و تلاشی برای شمارش، غیرمحتمل است.

یک نمونهٔ ساده از تجزیه، حالتی است که در آنجا عدد صحیح داده شده، تنها عوامل اول کوچکی دارد. برای مثال ۷۵۹۳۷۵ به سادگی تجزیه می‌شود چرا که می‌توانیم آن را به صورت $3 * 5^{5}$ بنویسیم. در رمزنگاری ما می‌خواهیم از اعداد صحیحی استفاده کنیم که تنها عوامل اول بزرگ داشته باشند. ترجیحاً یک عدد صحیح با دو عامل اول بزرگ انتخاب می‌شود، همان کاری که در سیستم رمزنگاری RSA انجام می‌شود.

اکنون یکی از بهترین الگوریتم‌های تجزیه، الگوریتم "غربال میدان اعداد" (NFS)[2] می‌باشدکه شامل یک فاز غربال و یک گام ماتریسی است. فاز غربال می‌تواند در بین تعداد زیادی از سیستم‌ها توزیع شود، اما گام ماتریسی نیاز به اجرا روی سوپرکامپیوترهای بزرگ دارد. سودمندی الگوریتم NFS برای اعداد صحیح خیلی بزرگ نمایان می‌شود (می‌تواند هرعدد صحیح از اندازهٔ 10^{150} را در چند ماه تجزیه کند. الگوریتم NFS زمان "زیرنمایی"[3] نیاز دارد (که هنوز خیلی کارآ نیست).

اثبات شناخته شده‌ای که ثابت کند تجزیهٔ عدد صحیح یک مسألهٔ NP-hard است و یا اینکه در زمان چندجمله‌ای، حل‌شدنی نمی‌باشد، موجود نیست. اگر هرمسألهٔ NP-hard در زمان چندجمله‌ای قابل حل بود، آنگاه تجزیه هم چنین خصوصیتی را دارا خواهد بود. اما امید خیلی کمی وجود دارد که این چنین باشد. در واقع، علم تجزیه پیشرفتهای کندی را در طی سالها دیده است و بزرگترین پیشرفتها از الگوریتم‌های پیشرفته بوجود می‌آید. تحت دانش فعلی باورکردنی است که بگوییم تجزیه در زمان چندجمله‌ای قابل حل نمی‌باشد.

[1] Factoring
[2] Number Field Sieve
[3] Sub-exponential

- "لگاریتم‌های گسسته"[1]. یک دستهٔ مهم دیگر از مسائل، مسألهٔ یافتن n با داشتن y است به طوریکه $y=g^n$. مسأله برای اعداد صحیح ساده است اما زمانیکه شرایط متفاوت باشد خیلی مشکل و سخت می‌شود.

برای محوکردن طبیعت n در g^n ، ما مجموعهٔ نامتناهی اعداد صحیح را در داخل یک مجموعهٔ متناهی از "کلاسهای باقیمانده"[2] تقسیم می‌کنیم. بطور قابل درک رشتهٔ اعداد صحیح گرفته شده روی یک دایره پیچانده می‌شود (که محیطی به طول m دارد).

اعداد ۰، m، ۲m، ۳m و ... همگی یک نقطه روی دایره را می‌پوشانند و بنابراین گفته می‌شود که دریک کلاس هم‌ارزی قرار دارند. (همچنین نوشته می‌شود: "(به پیمانهٔ m) ... = ۲m = m = ۰"). هرکلاس هم‌ارزی یک کوچکترین نماینده در ۰ ... m-۱ دارد. بنابراین هرعدد صحیح را می‌توان به صورت t + km نوشت، به ازای هر عدد صحیح t که ۰ <= t < m است. در این حالت نوشته می‌شود که n = t (mod m) که به m پیمانه (هنگ) گفته می‌شود. می‌تواند نشان داده شود که شخص می‌تواند با این کلاسهای اعداد صحیح (به پیمانهٔ m) جمع، تفریق و ضرب انجام دهد. این ساختارکه درآن m = p با اعداد اول p باشد، غالباً یک میدان اول یا یک "میدان گالوا"[3](GF(p)) نامیده می‌شود. زمانیکه m اول است، تقسیم کلاسهای صحیح غیرصفر، "خوش‌تعریف"[4] می‌باشد. بنابراین مسألهٔ لگاریتم گسسته در میدان متناهی GF(P) مطابق زیر بیان می‌شود: دو عدد صحیح غیرصفر مثبت a و g داده شده‌اند (هردو کوچکتر از p) ، محاسبه کنید n را به نحویکه $a = g^n \pmod p$. ما می‌توانیم g را به نحوی انتخاب کنیم که یک حل برای n به ازای هر a ی غیرصفر موجود باشد، به منظور سخت‌کردن این مسأله در رمزنگاری، p باید یک عدد اول بزرگ (درحدود $۱۰^{۳۰۰}$) و n هم بصورت کلی از همان مرتبه باشد.

درحال حاضر، این مسأله به سختی تجزیه در نظرگرفته می‌شود. بهترین شیوهٔ شناخته شده دراین زمان، "غربال میدان اعداد"[5] برای لگاریتم‌های گسسته می‌باشد (که از ایده‌های مشابه NFS برای تجزیه استفاده می‌کند). مسألهٔ لگاریتم گسسته ممکن است خیلی پیچیده‌تر از تجزیه عدد صحیح به نظر برسد، اما در چندین جنبه شبیه هستند. غالب ایده‌هایی‌که برای تجزیه عمل می‌کنند، همچنین می‌توانند در مورد لگاریتم‌های گسسته به کارگرفته شوند. امیدکمی برای یافتن یک الگوریتم با زمان چندجمله‌ای برای محاسبهٔ لگاریتم‌های گسسته در(GF(p موجود است.

مسألهٔ لگاریتم گسسته می‌تواند در محیط‌های زیاد دیگری، نظیر خمهای بیضوی به‌کار گرفته شود. یک دلیل برای این کاربرد این است که الگوریتم غربال میدان اعداد در اینجا کار نمی‌کند. روشهای دیگری برای محاسبهٔ لگاریتم‌های‌گسسته روی خمهای بیضوی وجود دارند اما به نظر می‌رسد که هنوز حل گسسته روی خمهای بیضوی

[1] Discrete logarithms
[2] Remainder class
[3] Galois Field
[4] Well defined
[5] Number field sieve

مشکل‌تر از روی GF(p) می‌باشد. این موضوع باعث تعدادی مزایای اندازهٔ کلید برای سیستم‌های رمزنگاری کلید عمومی بر پایهٔ خمهای بیضوی در مقابل سیستم‌های رمزنگاری بر پایهٔ تجزیه می‌گردد.

- کوله‌پشتی[1]. یک مجموعهٔ کوچک از اعداد صحیح داده شده‌است. مشخص‌کردن یک زیرمجموعه از چنین اعداد صحیح به نحویکه مجموعشان برابر عدد صحیح داده شده باشد، یک مسألهٔ کوله‌پشتی نامیده می‌شود. برای مثال، مجموعهٔ (۷، ۵، ۳، ۲) و ۱۰ داده شده‌اند. ما می‌توانیم به‌راحتی جواب ۱۰ = ۵ + ۳ + ۲ را بیابیم و بنابراین مسألهٔ کوله‌پشتی را بوسیلهٔ جستجوی جامع وکورکورانه(brute-force) حل کنیم. مسألهٔ کوله‌پشتی کلی به‌طور اثبات‌پذیرNP-hard می‌باشد و بنابراین ممکن است بهتر از تجزیه و لگاریتم‌گسستهٔ مورد استفاده در سیستم‌های رمزنگاری کلید عمومی به نظر برسد. متأسفانه، همهٔ سیستم‌های رمزنگاری که این ایده را زیربنا قرار داده‌اند؛ شکسته شده‌اند؛ چراکه نمونه‌ها یا حالات استفاده شده از مسأله واقعاً NP-hard نبوده‌اند.

- شبکه‌ها[2]. یک بردار پایهٔ $W_i = <w_1, w_2, ..., w_n>$ برای i از ۱ تا m تعریف می‌کنیم. شبکه با این پایه، تولید می‌شود؛ بدین معنی که عناصر شبکه به فرم $t_1W_1 + t_2W_2 + ... + t_mW_m$ می‌باشند که t_i ها صحیح هستند. مسألهٔ یافتن کوتاهترین بردار دریک شبکه (استفاده از فاصلهٔ اقلیدسی متداول) یک مسألهٔ NP- hard می‌باشد (برای شبکه‌هایی با بعد به اندازهٔ کافی بزرگ).

به هرحال، الگوریتم مشهور LLL بوسیلهٔ A.K. Lenstra، H.W. Lenstra و Jr. and L.Lovasz یک حل تقریبی را در زمان چندجمله‌ای محاسبه می‌کند. سودمندی الگوریتم LLL از این حقیقت برمی‌آید که درحالات زیادی، حل‌های تقریبی به خوبی کفایت می‌کنند و بطور شگفت‌آور، غالباً الگوریتم LLL دقیقاً کوتاهترین بردار را می‌دهد. این الگوریتم عموماً برای شکستن سیستم‌های رمزنگاری بر پایهٔ مسائل شبکه (Lattice) یا کوله‌پشتی مورد استفاده قرار گرفته است هرچندکه برای حمله علیه RSA وDSS[3] نیز به‌کار رفته است.

۴-۷-۶. سیستم‌های رمزنگاری کاربردی

میل گسترده در رمزنگاری کلید عمومی باعث ایجاد چندین سیستم رمزنگاری کاربردی مهم شده است. در ادامه، این الگوریتم‌ها به ترتیب "مسألهٔ زیربنایی"[4] لیست شده‌اند.

برای راهنمایی، یک سیستم رمزنگاری کلید عمومی ازیک مسألهٔ مشکل به طریق زیر ساخته می‌شود. یک مسألهٔ مشکل (برای مثال NP-hard) در نظر گرفته می‌شود به‌نحویکه می‌توان یک نمونه از آن را یافت که در زمان چندجمله‌ای قابل حل است. برای رمزکردن یک پیام، پیام را به چنین نمونه‌ای از مسألهٔ مشکل تغییر شکل می‌دهند. سپس از کلید عمومی

[1] Knapsack
[2] Lattices
[3] Digital Signature Standard
[4] Underlying problem

برای تبدیل مسألهٔ ساده به یک نمونهٔ مشکل استفاده می‌گردد. حاصل سپس از طریق یک کانال (ناامن) به گیرنده فرستاده می‌شود. برای رمزگشایی از کلید خصوصی برای تبدیل مسألهٔ مشکل به نوع ساده استفاده می‌شود تا پیام قابل بازیافت گردد. همهٔ سیستم‌های کلید عمومی از این قاعده استفاده می‌کنند؛ اگرچه در جزئیات می‌توانند به‌شدت با هم متفاوت باشند (نظیر مسألهٔ زیربنایی، یا ساختار کلید عمومی و خصوصی).

برای بررسی خوب روی طول کلیدهای مناسب باید مقالهٔ Lenstra و Verheul را با عنوان " Selecting Cryptographic Key Sizes" مطالعه نمود که در مبحث طول کلید از بخش تکنیکهای رمزنگاری به آن اشاره شد. درادامه به همراه هر سیستم رمزنگاری، پیشنهادهای فعلی برای اندازهٔکلید هرجا که مناسب باشد، خواهد آمد. این توصیه‌ها همیشه با پیشنهاد Lenstra و Verheul یکسان نمی‌باشند.

۴-۷-۶-۱. تجزیه : RSA و Rabin

◆ RSA[1]

RSA الگوریتم کلید عمومی است که بسیار مورد استفاده واقع شده است. این الگوریتم توسط ریاضیدانان MIT، Rivest، Shamir و Adleman در سال ۱۹۷۸ بوجود آمد. این الگوریتم می‌تواند هم برای رمزکردن و هم برای امضاهای دیجیتالی و رمزکردن بلوکهای کوچک داده مورد استفاده قرار گیرد. امنیت RSA بطور معمول، هم‌ارز تجزیه درنظر گرفته می‌شود، اگرچه این مطلب ثابت نشده است.

محاسبهٔ RSA با اعداد صحیح هم نهشت $n = p*q$ ، برای دو عدد اول مخفی بزرگ p و q انجام می‌شود. برای رمزکردن، پیام m ، با یک توان عمومی کوچک e به‌توان رسانده می‌شود. برای رمزگشایی، گیرندهٔ Ciphertext $c = m^e \pmod n$ ، وارون ضربی $d = e^{-1} \pmod{(p-1)(q-1)}$ را محاسبه می‌کند (برای وجود وارون، e باید به‌طور مناسبی انتخاب شود) و بدست می‌آورد که $c^d = m^{e*d} = m \pmod n$. کلید خصوصی شامل n ، p ، q ، e و d (و p و q) می‌توانند فراموش شوند) و کلید عمومی تنها حاوی n و e می‌باشد. مسألهٔ ایجاد شده برای حمله‌کننده، محاسبهٔ d معکوس e است که فرض می‌شود ساده‌تر از تجزیهٔ n نباشد. RSA از کلید و بلوک با طول متغیر استفاده می‌کند.

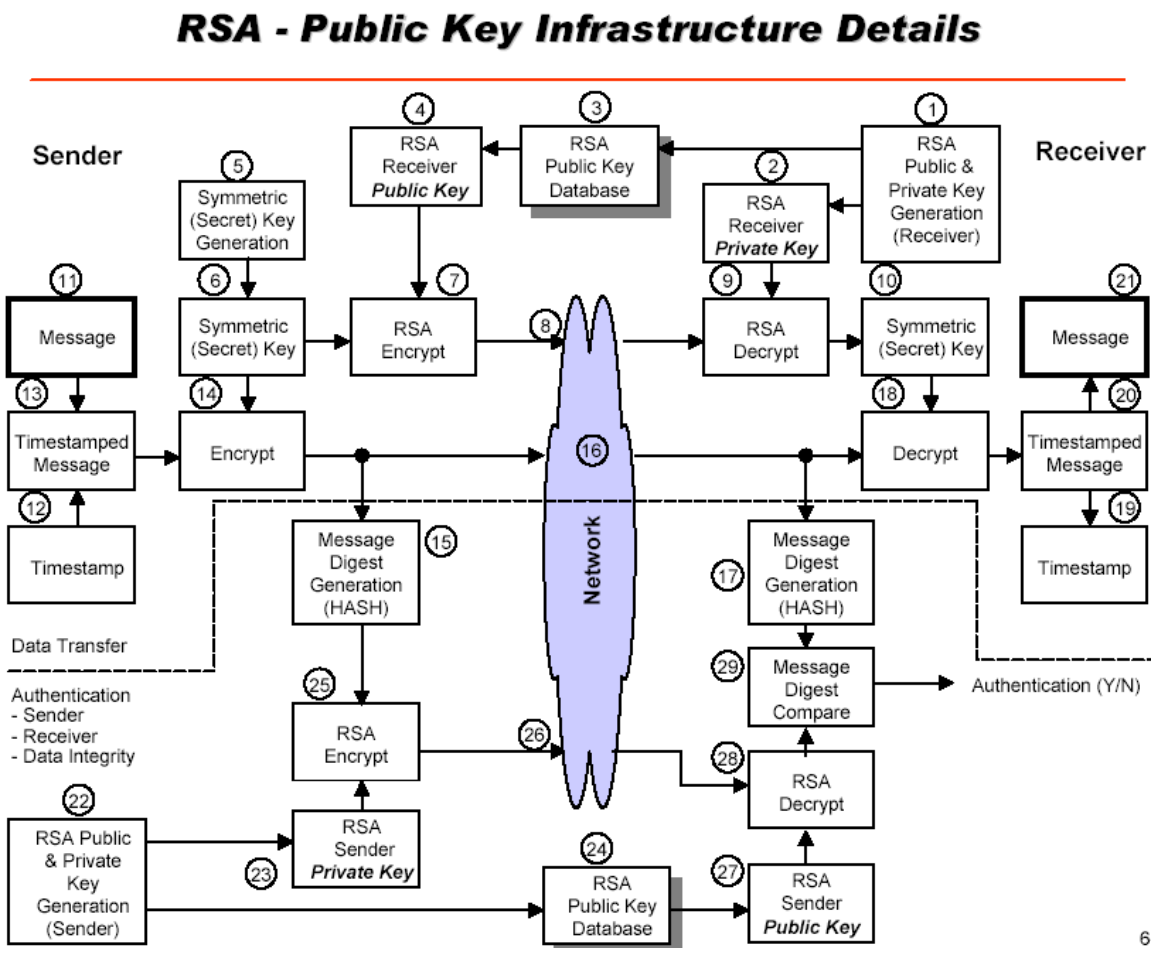
اندازهٔ کلید (اندازهٔ پیمانه) باید بزرگتر از ۱۰۲۴ بیت باشد (بدین معنی که باید از اندازهٔ $۱۰^{۳۰۰}$ باشد). برای یک حاشیهٔ امنیت معقول کلیدهایی با اندازهٔکلید ۲۰۴۸ بیت امنیتی را برای دهه‌ها مهیا می‌کنند. پیشرفت‌های مهیج در تجزیهٔ اعداد صحیح بزرگ ممکن است RSA را آسیب‌پذیرکند. توانایی کامپیوترها برای تجزیهٔ اعداد بزرگ و بنابراین حمله به روشهایی نظیر RSA ، به‌سرعت درحال پیشرفت است و سیستم‌های امروزه می‌توانند عوامل اول اعدادی با بیش از ۱۴۰ رقم را بیابند. به هرحال، محافظت فرض‌شده از RSA این است که کاربران بسادگی می‌توانند اندازهٔ کلید را افزایش دهند تا همیشه از منحنی

[1] Rivest-Shamir-Adlman

پردازش کامپیوتر جلوتر باشند. پیاده‌سازیهای خوب از افزونگی[1](یاpadding با ساختار مشخص) به ترتیبی که از حملات با استفاده از ساختار افزایشی (ضربی) Ciphertext جلوگیری می‌کند، استفاده می‌کنند. RSA در برابر حملات chosen plaintext وhardware and fault attacks آسیب‌پذیر است (این حملات در گزارشهای بعدی توضیح داده می‌شوند). همچنین حملات مهمی علیه نماهای خیلی کوچک و "تجزیهٔ پاره‌ای نشان داده شده از پیمانه‌ها"[2] موجودند. پیاده‌سازی‌های مناسب از الگوریتم RSA با افزونگی در استانداردهای PKCS (RFC های ۲۳۱۴، ۲۳۱۵ و ۲۴۳۷) به‌خوبی بیان شده است. این استانداردها توصیف با جزئیات دربارهٔ چگونگی پیاده‌سازی رمزکردن و امضاهای دیجیتالی به‌علاوهٔ فرمت‌هایی برای ذخیرهٔ کلیدها را شامل می‌شوند. الگوریتم RSA ساده[3] نباید در هیچ کاربردی استفاده شود. استفاده از پیاده‌سازی‌هایی که از استانداردها پیروی می‌کنند توصیه می‌شود چراکه این کار همچنین مزیت اضافی توانایی همکاری با اغلب قراردادهای عمده را دارد.

RSA اکنون الگوریتم کلید عمومی است که دارای بیشترین اهمیت می‌باشد. این الگوریتم در ایالات‌متحده و کانادا دارای حق امتیاز بود که در سپتامبر ۲۰۰۰ به اتمام رسید اما به نظر نمی‌رسد این موضوع در محبوبیت RSA اثرسوئی داشته باشد. جزئیات زیرساختار RSA مورد استفاده در حال حاضر در شکل ۴-۳۴ نشان‌داده شده است.

[1] Redundancy
[2] Partially revealed factorization of the modulus
[3] Plain



شکل ۴- ۳۴ جزئیات زیرساختار کلید عمومی RSA

- ***چند نکته در مورد انتخاب p و q***

- p و q باید طوری انتخاب شوند که n = pq بطور محاسباتی تجزیه نشود. محدودیت مهم روی p و q به‌خاطر جلوگیری از الگوریتم تجزیهٔ خم بیضوی این است که p و q باید حدوداً یک طول داشته باشند و بطور مؤثر بزرگ باشند. برای مثال، اگر یک پیمانهٔ n ۱۰۲۴ بیتی قرار است که استفاده شود، آنگاه هرکدام از p و q ها باید حدود ۵۱۲ بیت طول داشته باشند.

- محدودیت دیگر در انتخاب اعداد اول p و q آن است که اختلاف p-q نباید خیلی کوچک باشد. اگر p-q کوچک باشد آنگاه p≈q و درنتیجه p≈√n، بنابراین n می‌تواند بطور مؤثری به‌سادگی با تقسیم آزمایشی برهمهٔ اعداد صحیح فرد نزدیک به √n تجزیه شود. اگر p و q به صورت تصادفی انتخاب شوند، آنگاه احتمالاً p-q به صورت مناسبی بزرگ خواهد بود.

- علاوه بر این محدودیتها، افراد زیادی توصیه کرده‌اند که p و q باید اعداد "اول قوی"[1] باشند. یک عدد اول p گفته می‌شود که یک عدد اول قوی است اگر سه شرط زیر را برآورده کند:
  - p-۱ یک عامل اول بزرگ داشته باشد که با r نشان داده می‌شود.
  - p+۱ یک عامل اول بزرگ داشته باشد.
  - r-۱ یک عامل اول بزرگ داشته باشد.

RSA در استانداردهای پیشنهادی منتشرشدهٔ زیادی در سراسر جهان ترکیب شده است. همچنین در بسیاری از قراردادهای اینترنتی نظیر S/MIME ، S-HTTP و SSL به‌کار رفته است. در مورد محصولات تجاری هم به گونه‌های مختلفی چه محصولات سخت‌افزاری و چه نرم‌افزاری زیادی استفاده شده است.

مخترعین RSA در زمانهای مختلف "مبارزه‌طلبی"[2] را همراه پاداش برای تجزیه انجام داده‌اند (یک نمونه درسال ۱۹۹۷ با پاداش ۱۰۰ دلار). RSA Laboratories درحال حاضر، اندازهٔ کلیدهای ۱۰۲۴ بیتی را برای استفادهٔ شرکتی و ۲۰۴۸ بیتی را برای کلیدهای خیلی با ارزش نظیر جفت کلید root (ریشه) مورد استفاده در یک منبع CA توصیه می‌کند. اطلاعات با ارزش کمتر می‌توانند با کلیدهای ۷۸۶ بیتی نیز رمز شوند.

در بعد سخت افزاری، RSA می‌تواند در تلفنهای امن، کارتهای شبکهٔ Ethernet و کارتهای هوشمند استفاده شود. RSA در سیستم‌های عامل Windows، Apple، Sun و Novell استفاده می‌شود و به عنوان یک استاندارد در صنعت بانکداری عمل می‌کند.

- ***استاندارد PKCS #1***

این سند، توصیه‌هایی برای پیاده‌سازی‌های رمزنگاری کلید عمومی برپایهٔ الگوریتم RSA، شامل بر مفاهیم Primitive های رمزنگاری، روشهای رمزکردن، ... و چگونگی نمایش کلیدها می‌باشد. RSA Data Security آخرین نسخهٔ آن را که 2.01 می‌باشد در تاریخ ۲۰۰۲/۶/۱۴ نهایی کرده‌است.

- **سیستم رمزنگاری Rabin**

این سیستم رمزنگاری می‌تواند یک خویشاوند RSA درنظر گرفته شود، اگرچه‌که یک فرآیند رمزگشایی خیلی متفاوت دارد. این سیستم رمزنگاری در سال ۱۹۷۹ ارائه شده است. آنچه که آن را جالب می‌سازد این است که شکستن Rabin بطور اثبات‌پذیری هم‌ارز تجزیه می‌باشد. (اولین نمونه‌ای بودکه بطور اثبات‌پذیر بررسی شد).

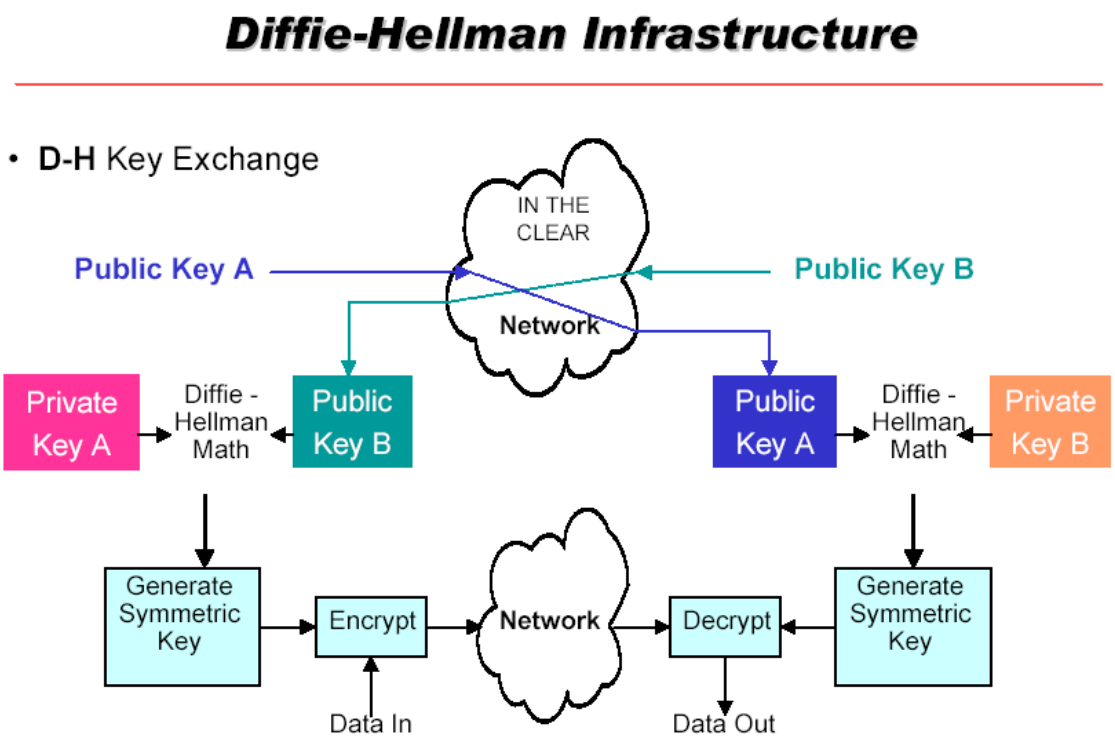
[1] Strong Prime
[2] Challenge

Rabin از نمای ۲ (یا هرعدد صحیح زوج) به جای اعداد صحیح فرد که در RSA استفاده می‌شود، استفاده می‌کند. این عمل دو نتیجه دارد. اول، ثابت می‌شود که هم‌ارز تجزیه می‌باشد؛ دوم، اینکه رمزگشایی، - حداقل دربعضی موارد- خیلی مشکل می‌شود. نتیجهٔ دوم به‌خاطر مسائل در تصمیم‌گیری (قطعی‌کردن) اینکه کدام یک از نتایج ممکن از فرآیند رمزگشایی صحیح می‌باشد، حاصل می‌شود. از آنجائیکه هم‌ارز تجزیهٔ پیمانه می‌باشد، اندازهٔ پیمانه مهمترین پارامتر امنیت می‌باشد. پیمانه‌های بزرگتر از ۱۰۲۴ بیت فرض می‌شوند که هنوز امن می‌باشند. درحال حاضر استانداردهایی برای الگوریتم Rabin موجود نیست اما الگوریتم مزبور در چندین کتاب شرح داده شده‌است. پروژهٔ IEEE P1363 ممکن است یک استاندارد پیشنهاد دهد و بنابراین آن را پرکاربرد نماید. رمزکردن Rabin عمل به‌شدت سریعی است چراکه تنها شامل یک "مربع‌کردن پیمانه‌ای"[1] می‌باشد. در مقایسه، رمزکردن RSA با e = ۳ یک ضرب پیمانه‌ای ویک مربع‌کردن پیمانه‌ای طول می‌کشد. رمزگشایی Rabin کندتر از رمزکردن آن است، اما با رمزگشایی RSA قابل مقایسه است.

۴-۷-۶-۲. لگاریتم‌های گسسته : Diffie-Hellman ، ElGamal و DSS.

◆ Diffie-Hellman

الگوریتم Diffie-Hellman یک قرارداد معمول برای استفاده در مبادلهٔ کلید می‌باشد (شکل ۴-۳۵).



شکل ۴-۳۵ زیرساختار Diffie-Hellman

[1] Modular squaring

درقراردادهای رمزنگاری زیادی دو طرف قصد شروع ارتباط را دارند. فرض کنید آنها در ابتدا هیچ رمز مشترکی ندارند و بنابراین نمی‌توانند از سیستم‌های رمزنگاری کلید متقارن استفاده کنند. مبادلهٔ کلید با قرارداد Diffie-Hellman این مشکل را حل می‌کند. این عمل برپایهٔ یک مسألهٔ مربوط به لگاریتم‌های گسسته به نام مسألهٔ Diffie-Hellman بنا شده است. این مسأله سخت و مشکل در نظرگرفته می‌شود و در بعضی نمونه‌ها به سختی مسألهٔ لگاریتم گسسته می‌باشد.

قرارداد Diffie-Hellman به طور معمول در صورتیکه از یک گروه ریاضی مناسب استفاده شود، امن در نظرگرفته می‌شود. مشخصاً، عنصر مولد مورد استفاده در به توان رسانی‌ها باید یک دورهٔ بزرگ (یا مرتبهٔ بزرگ) داشته باشد.

الگوریتم‌های لگاریتم‌گسسته می‌توانند به همراه حملات منفعل که درحال حاضر بهترین حملات ممکن - با فرض پارامترهای به خوبی انتخاب شده- می‌باشند، برای حمله به Diffie-Hellman استفاده شوند. اگر Diffie-Hellman با ریاضیات معمولی به پیمانهٔ یک عدد اول به‌کارگرفته شود، کافیست یک عدد اول به اندازهٔ کافی بزرگ انتخاب شود و مقداری مراقبت در انتخاب "عنصرمولد"[1] لحاظ شود. مسائل ضعیف ممکن است از انتخاب بد مولد ایجاد شده باشند. حملات علیه Diffie-Hellman همچنین شامل بر حملهٔ man-in-the-middle (واسطه) می‌شود. این حمله نیاز به "مداخلهٔ وفقی"[2] دارد، اما درعمل اگر قرارداد از اقدامات متقابلی نظیر امضاهای دیجیتالی استفاده نکند، به‌سادگی قابل انجام است. بطور معمول Diffie-Hellman روی سخت‌افزار پیاده‌سازی نمی‌شود و بنابراین حملات سخت‌افزاری تهدید مهمی نیستند. البته این وضع ممکن است در آینده تغییرکند (زمانیکه "ارتباطات سیار"[3] بیشتر گسترش می‌یابد).

◆ DSS[4] (استاندارد امضای دیجیتالی)

DSS مکانیزمی است که تنها برای امضا مورد استفاده است و توسط دولت آمریکا انتخاب شده است. NIST، DSS را ایجادکرد تا برای تولید امضای دیجیتالی و تأیید آن استفاده شود، یکی از شرایط برای ایجاد آن بدون حق امتیاز بودن آن، بود. الگوریتم زیربنایی DSA[5] شبیه الگوریتمی است که بوسیله ElGamal یا بوسیلهٔ الگوریتم امضای Schnorr استفاده می‌شود. همچنین به‌اندازهٔ کافی کارآ و مؤثر می‌باشد، هرچندکه برای تأیید امضا به کارآیی RSA نیست. این استاندارد بیان می‌کند که DSS منحصراً از SHA-1 برای محاسبهٔ "خلاصه پیام‌ها" استفاده کند.

مشکل اصلی در DSS اندازهٔ ثابت زیرگروه (مرتبهٔ عنصر مولد) می‌باشد که امنیت را درحدود تنها ۸۰ بیت محدود می‌کند. حملات سخت‌افزاری می‌تواند یک نگرانی نسبت به بعضی پیاده‌سازی‌های DSS به حساب آید. به هرحال، DSS به‌صورت خیلی گسترده‌ای استفاده می‌شود و به عنوان یک الگوریتم خوب پذیرفته شده است. این در حالی است که افراد زیادی

[1] Generator element
[2] Adaptive Intervention
[3] Mobile Communication
[4] Digital Signature Standard
[5] Digital Signature Algorithm

مشکلات بالقوه‌ای درآن یافته‌اند. برای مثال، نفوذ (تراوش) داده به امضاء و فاش شدن کلید خصوصی اگر دو پیام مختلف با استفاده از یک عدد تصادفی امضاء شده باشند، بعضی از مشکلاتی هستند که پیش می‌آیند.

◆ رمزکنندهٔ کلید عمومی ElGamal

توسط طاهرالجمال رمزنگار مصری‌الاصل ارائه شده است که ایشان جزء ایجادکنندگان قرارداد SSL در شرکت Netscape می‌باشند. این الگوریتم یک توسعه (بسط) سرراست از ایدهٔ اصلی Diffie/Hellman برای تولید رمز مشترک می‌باشد. این الگوریتم اساساً یک رمز اشتراکی را تولید می‌کند که از آن به عنوان یک one-time pad برای رمزکردن یک بلوک داده استفاده می‌شود. ElGamal جد (سلف) DSS می‌باشد و امروزه به خوبی قابل استفاده است، هرچند که استاندارد شناخته شدهٔ معروفی برای آن ایجاد نشده است.

- ***نکاتی در مورد کارآیی رمزکردن ElGamal:***

- دو توان‌رسانی پیمانه‌ای آن یعنی $\alpha^k \bmod p$ و $(\alpha^a)^k \bmod p$ با انتخاب نماهای تصادفی k که دارای مقداری ساختار اضافی است، می‌تواند سریعتر انجام شود. باید دقت شود که تعداد نماهای ممکن به اندازهٔ کافی بزرگ باشد تا مانع جستجوی الگوریتمی مانند giant-step baby-step شود.

- یک بدی رمزکردن ElGamal این است که یک بسط پیام با عامل ۲ دارد بدین معنی که ciphertext از لحاظ طولی دو برابر plaintext متناظر می‌باشد.

- امنیت رمزکردن ElGamal

  - بر پایهٔ مسألهٔ لگاریتم گسسته در Z_p^* است؛ اگرچه که چنین هم ارزی ثابت نشده است.

  - لزوماً باید اعداد صحیح تصادفی مختلف k برای رمزکردن پیامهای مختلف مورد استفاده قرار گیرد.

- ***اندازهٔ پارامتر توصیه شده***

پیمانهٔ ۵۱۲ بیتی p فقط امنیت حاشیه‌ای از حملهٔ مجتمع را فراهم می‌کند. از سال ۱۹۹۶، یک پیمانهٔ p حداقل ۷۶۸ بیتی توصیه شده‌است. برای امنیت با زمان طولانی، ۱۰۲۴ بیت یا بیشتر باید استفاده شود.

◆ سیستم‌های رمزنگاری خم بیضوی (ECC)[1]

این سیستمها عملاً راه دیگری برای پیاده‌سازی روشهای لگاریتم گسسته می‌باشند. درسال ۱۹۸۵، "رمزنگاری خم بیضوی" (ECC)[1]، یک روش PKC بر پایهٔ الگوریتمی کاملاً متفاوت، به صورت مستقل توسط رمزنگارانی به نامهای Victor

[1] Elliptic Curve cryptosystems

Miller (IBM) و Neal Koblitz (دانشگاه واشنگتن) پیشنهاد شد. مسألهٔ به‌طور مشخص مورد تمایل برای PKC، "مسألهٔ لگاریتم گسسته خم بیضوی" می‌باشد. شبیه مسألهٔ تجزیه (به عاملهای اول)، این هم یک مسألهٔ "مشکل" دیگر می‌باشد (که در بیان به‌طور گول‌زننده‌ای ساده است!) : دو نقطهٔ P و Q روی یک خم بیضوی داده شده‌اند؛ عدد صحیح i را درصورت وجود بیابید به طوری که P = iQ.

خمهای بیضوی از نظریهٔ اعداد و هندسهٔ جبری (تحلیلی) استفاده می‌کنند. این خمها می‌توانند روی هرمیدانی از اعداد (یعنی حقیقی، صحیح و مختلط) تعریف شوند، هرچندکه معمولاً دیده می‌شود که روی میدانهای متناهی برای کاربردهایی در رمزنگاری استفاده می‌شوند. یک خم بیضوی مشتمل است بر مجموعهٔ اعداد حقیقی (x,y) که در معادلهٔ زیر صدق می‌کنند.

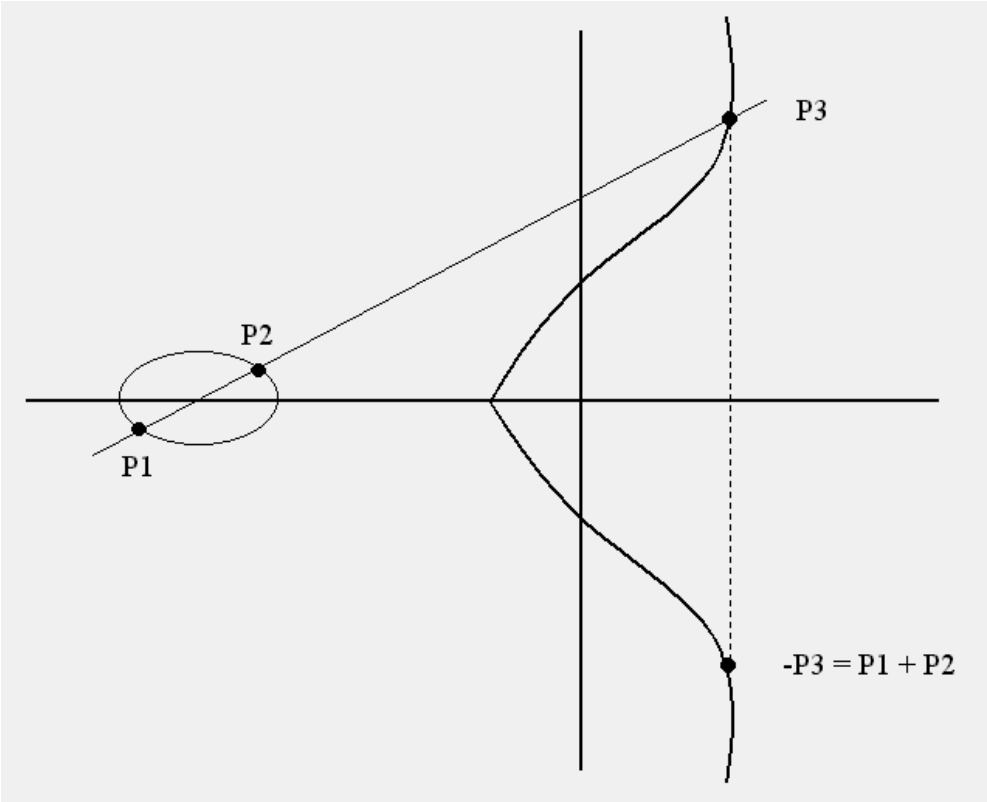
$$y^2 = x^3 + ax + b$$

مجموعهٔ تمام جواب‌های معادله، خم بیضوی را تشکیل می‌دهند. تغییر a و b شکل خم را تغییر می‌دهد و تغییرات کوچک در این پارامترها می‌تواند منجر به تغییرات قابل توجهی در مجموعهٔ جوابهای (x,y) شود.

شکل ۴-۳۶ جمع دو نقطهٔ روی یک خم بیضوی را نشان می‌دهد که برطبق مجموعه‌ای از قوانین ساده تعریف می‌شود. نقطهٔ P1 به اضافهٔ نقطهٔ P2 مساوی نقطهٔ (x,-y) = P3- است که (x,y) = P3 و P3 نقطهٔ تلاقی خم بیضوی و خط گذرنده از نقاط P1 و P2 می‌باشد. همانطور که نشان داده شده است، تغییرات کوچک در P1 یا P2 می‌تواند باعث تغییر زیاد در مکان P3 گردد.

نقطهٔ P3 به‌عنوان مضربی از نقطهٔ شروع P1 محاسبه می‌شود یا به عبارت دیگر P3 = nP1. یک حمله‌کننده ممکن است P1 وP3 را بداند اما یافتن عدد صحیح n برای حل، مسألهٔ مشکلی می‌باشد. P3،کلید عمومی است و بنابراین n، کلید خصوصی می‌باشد.

[1] Elliptic Curve Cryptography



شکل ۴- ۳۶ جمع خم بیضوی

همان‌طورکه در قبل گفته شد، نقاط روی خمهای بیضوی می‌توانند با هم جمع شوند و یک ساختار به نام گروه (درحقیقت گروه آبلی) را تشکیل دهند. این دقیقاً یک راه بیان این مطلب است که با آنها نیز می‌توان همانند اعداد صحیح عملیات ریاضی انجام داد زمانیکه تنها جمع و تفریق مورد نظر باشند. در خصوص رمزنگاری، خمهای بیضوی مزایای نظری زیادی دارند، همچنین خیلی عملی نیز هستند. در اغلب خمهای "ماورای بیضوی"[1]، الگوریتم زیرنمایی شناخته شده‌ای برای محاسبهٔ لگاریتم‌های گسستهٔ نقاط خمهای بیضوی برخلاف لگاریتم‌های گسسته در (گروه ضربی) یک میدان متناهی، وجود ندارد. یک مزیت عملی از عدم وجود یک الگوریتم محاسبهٔ لگاریتم‌گسستهٔ سریع برای خمهای بیضوی، کاهش طول کلید است. همچنین امضاهای دیجیتالی تولید شده و پیامهای رمزشده کوچک هستند. در واقع، یک راه خیلی سادهٔ محاسبهٔ حد امنیت برای طول کلید، در نظرگرفتن اندازهٔ کلید برای یک سیستم رمزنگاری کلید محرمانه به بیت و سپس ضرب آن در ۲ می‌باشد. این عدد تخمینی بدست می‌دهد، که در این زمان برای نمونه‌ای کلی از خمهای بیضوی مناسب است.

خمهای بیضوی می‌توانند به‌طور خیلی مؤثر و کارآ در سخت‌افزار و نرم‌افزار پیاده‌سازی شوند و درسرعت نیز به خوبی با سیستم‌های رمزنگاری نظیر RSA و DSS رقابت می‌کنند. RSA برای بیش از دو دهه نقطهٔ اتکای PKC بوده است، اما ECC به خاطر پتانسیل فراهم آوردن سطوح امنیت مشابه در مقایسه با RSA با کلیدهای بسیار کوتاهتر تحریک‌آمیز و مهیج می‌باشد. به همین دلیل، اخیراً توجهات زیادی در صنعت و مراکز دانشگاهی به آن شده است. همچنین چندین تقلا

[1] Hyperelliptic curves

برای استانداردسازی سیستم‌های رمزنگاری خم بیضوی انجام شده است (برای مثال، ECDSA بوسیلهٔ ANSI). درحال حاضر خمهای بیضوی به‌طور گسترده‌ای مشهور هستند، اما در عمل خیلی پرکاربرد نشده‌اند. امنیت سیستم‌های رمزنگاری خم بیضوی نسبتاً برای مدتها پایدار بوده است، اگرچه پیشرفتهای قابل توجهی در حملات علیه نمونه‌های خاصی حاصل شده‌اند. هرچند که اینها بوسیلهٔ محققین پیشرو در چندین سال قبل حدس‌زده شده بود و تعجب زیادی را بر نیانگیخت.

الگوریتم XTR که اخیراً بوسیلهٔ Lenstra و Verheul ارائه شده است می‌تواند یک رقابت‌کنندهٔ خوب برای خمهای بیضوی باشد. هرچندکه، به‌نظر می‌رسد خمهای بیضوی اندکی در کارآیی بهتر باشند و مشخصاً تناسب بهتری در اندازهٔ کلید داشته باشند.

Certicom Corp. (http://www.certicom.com/) که یکی از طرفداران واقعی ECC می‌باشد (این شرکت تحقیقات مهمی روی پیاده‌سازی کارتهای خود با استفاده از ECC انجام داده‌است)، رابطهٔ اندازهٔ کلید بین ECC و RSA را برطبق جدول ۴-۱۷ پیشنهاد می‌دهد.

جدول ۴-۱۷ مقایسهٔ کلید ECC و RSA (اندازهٔ کلید به بیت)

نسبت اندازهٔ کلید RSA: ECC	اندازهٔ کلید ECC	زمان برای شکستن کلید (MIPS سال[۱])	اندازهٔ کلید RSA
۵:۱	۱۰۶	$۱۰^{۴}$	۵۱۲
۶:۱	۱۳۲	$۱۰^{۸}$	۷۶۸
۷:۱	۱۶۰	$۱۰^{۱۱}$	۱۰۲۴
۱۰:۱	۲۱۰	$۱۰^{۲۰}$	۲۰۴۸
۳۵:۱	۶۰۰	$۱۰^{۷۸}$	۲۱۰۰۰

RSA Inc. در دفاع از RSA بیان می‌کند که ECC برای امضاء و رمزگشایی از RSA سریعتر می‌باشد، اما برای تأیید امضاء و رمزکردن کندتر است.

استانداردهای زیادی ECC را پذیرفته‌اند:

- *ANSI X9 (مانند ECDSA)*

[۱] یک "MIPS سال" مساوی است با : کارکردن یک کامپیوتر با توان پردازشی ۱ میلیون دستورالعمل در ثانیه به‌مدت ۱ سال معادل $۳,۱۵۳۶ * ۱۰^{۱۳}$ دستورالعمل.

- *ATM Forum (برای شبکه‌های Asynchronous Transfer Mode)*
- *NIST FIPS (FIPS 186-2 که شامل ECDSA می‌باشد)*
- *IEEE P1363*
- *IETF*[1]
- *ISO/IEC SC27*
- *SECG*[2]
- *WTLS*[3]

در ادامه، مقایسهٔ طول آیتمهای رمزشده بوسیلهٔ ECC و RSA/DSA آورده شده است:

- کلید کوچک : ۱۶۱ بیت ECC حدوداً هم‌ارز ۱۰۲۴ بیت RSA/DSA
- امضای کوچک : ۳۲۲ بیت ECDSA حدوداً هم‌ارز ۱۰۲۴ بیت RSA
- "گواهی کوچک"[4] (کلید و امضا به بایت) : ECDSA:۶۲ ، DSA:۱۶۸، RSA:۲۵۶

همچنین جدول ۴-۱۸، مقایسهٔ دقیقتری را از لحاظ اندازهٔ کلید مهیا می‌کند.

جدول ۴-۱۸ مقایسهٔ طول کلید (اعداد به بیت نشان‌داده شده‌اند)

کلید متقارن	۸۰	۱۱۲	۱۲۸	۱۹۲	۲۵۶
ECC n	۱۶۱	۲۲۴	۲۵۶	۳۸۴	۵۱۲
DSA q	۱۶۰	۲۲۴	۲۵۶	۳۸۴	۵۱۲
DSA p	۱۰۲۴	۲۰۴۸	۳۰۷۲	۷۶۸۰	۱۵۳۶۰
RSA n	۱۰۲۴	۲۰۴۸	۳۰۷۲	۷۶۸۰	۱۵۳۶۰

نتیجه : طول کلید RSA/DSA برای افزایش امنیت ارائه‌شده نسبت به ECC رشد خیلی سریعتری دارد.

بعضی مزایای داشتن اندازهٔ کلید کوچکتر، محاسبات سریعتر، کاهش در توان پردازشی مورد نیاز، فضای ذخیره سازی و پهنای باند باعث می‌شود که ECC برای محیط‌های محدودی نظیر Pager ها و PDAها، تلفنهای سلولی و کارتهای هوشمند ایده‌آل باشد. ازطرفی پیاده‌سازی ECC نیاز به انتخاب چندین گزینه نظیر نوع میدان متناهی زیرلایه، الگوریتمهایی برای پیاده‌سازی عمل گروه بیضوی و قراردادهای خم بیضوی دارد. تعداد زیادی از این انتخابها، تأثیر قابل‌توجهی روی کارآیی کلی دارد.

[1] Internet Engineering Task Force
[2] Standard for Efficient Cryptography Group
[3] Wireless Transport Layer Security
[4] Short certificate

◆ LUC

یک سیستم رمزنگاری کلید عمومی است که از یک گروه ویژه برپایهٔ "دنباله‌های لوکاس"[1] (مرتبط با سریهای فیبوناچی) به عنوان بلوک ساختمانی پایه استفاده می‌کند. LUC می‌تواند همهٔ الگوریتم‌های برپایهٔ لگاریتم‌های گسستهٔ متداول را پیاده‌سازی کند و از یک جهت LUC یک کلاس از الگوریتم‌های کلید عمومی می‌باشد. ممکن است که ساختار زیربنایی الگوریتم به عنوان یک گروه ضربی مشخص از میدان متناهی مشخصهٔ p با درجه ۲ (به صورت F_{p2} نوشته می‌شود) در نظر گرفته شود و این موضوع می‌تواند برای اثبات وجود یک الگوریتم زیرنمایی برای محاسبهٔ لگاریتم‌های گسسته و در نتیجه حمله به LUC استفاده شود. بنابراین ممکن است اینطور به‌نظر برسد که نسبت به LUC تمایل کمی وجود دارد چراکه LUC فقط یک راه دیگر برای پیاده‌سازی الگوریتم‌های برپایهٔ لگاریتم گسسته بر روی میدان‌های متناهی می‌باشد. به هرحال، LUC از عملیات ریاضی مشخص مشتق شده از دنباله‌های لوکاس استفاده می‌کند که ممکن است به اندازهٔ یک ضریب ثابت از راه مستقیم‌تر، سریعتر باشد.

الگوریتم‌های کلید عمومی مختلف برپایهٔ ریاضیLUC، LUCDIF (LUC Diffie-Hellman)، LUCELG (LUC ElGamal) و LUCDSA (LUC Digital Signature Algorithm) می‌باشند که تعداد زیادی از اینها دارای حق‌امتیاز می‌باشند.

مقادیر مورد استفاده در الگوریتم‌های LUC می‌توانند به عنوان یک جفت از مقادیری که تعدادی مزیت اضافی درمقابل استفادهٔ عینی از اعداد صحیح به پیمانهٔ p دارند، درنظرگرفته شوند. محاسبات تنها شامل اعدادی می‌شود که نیاز به نیمی از بیتهایی دارند که در حالت استفاده از اعداد صحیح به پیمانهٔ p نیاز است. از آنجایی که عملیات گروه LUC برای محاسبه ساده است، این امر عاملی است که الگوریتم‌های LUC را با RSA و DSS قابل مقایسه کند. به هرصورت، درحال حاضر به‌نظر می‌رسد که دلیل مهمی برای استفاده از سیستم‌های رمزنگاری LUC وجود ندارد، چراکه آنها مزیت کمتری نسبت به خمهای بیضوی یا XTR ارائه می‌دهند.

◆ XTR

یک سیستم رمزنگاری کلید عمومی است که توسط Arjen Lenstra و Eric Verheul در آمریکا ایجاد و در سال ۲۰۰۰ منتشر شده است. XTR تلفظ کلمهٔ ECSTR نمایندهٔ عبارت Efficient Compact Subgroup Representation Trace می‌باشد. XTR یک سیستم رمزنگاری جدید نیست بلکه یک سیستم برپایهٔ زیرگروه سنتی مسألهٔ لگاریتم گسسته می‌باشد که تنها برپایهٔ یک روش جدید برای نمایش عناصر یک زیرگروه از گروه ضربی یک میدان متناهی بنا شده است و روش کلاسیک Diffie-Hellman را کارآتر و فشرده‌تر می‌کند. XTR شبیه LUC است درآنجا که از یک گروه ضربی مشخص از یک میدان متناهی مشخص (درواقع F_{p6}) به عنوان گروه زیربنایی (زیرساختاری) خود استفاده می‌کند. به

[1] Lucas Sequences

هرحال، XTR خصوصیات جدیدی نظیر نیاز به فقط حدود یک سوم از بیتهایی که برای امضاها و پیامهای رمزشده استفاده می‌شود، دارد. این مزیت نتیجهٔ استفاده از "نمایش اثر"[1] مشخص از عناصر این گروه و انجام همهٔ محاسبات با استفاده از این نمایش می‌باشد. همهٔ الگوریتم‌های کلید عمومی برپایهٔ لگاریتم‌های گسسته می‌توانند با ایده‌های XTR پیاده‌سازی شوند. بنابراین به یک طریق، XTR یک نام عمومی برای یک کلاس از الگوریتم‌های کلید عمومی همانند LUC می‌باشد.

در صورت محرز شدن کارایی شگفت‌انگیز XTR و بر طبق گفتهٔ Lenstra و Verheul ممکن است که این الگوریتم یک جانشین خوب برای خمهای بیضوی، DSS و حتی RSA در کاربردهایی نظیرSSL/TLS، کارتهای هوشمند کلید عمومی، WAP/WTLS[2] ، IPSec/IKE[3] و SET[4] باشد، چراکه با اندازهٔ کلیدهای عمومی خیلی کوچکتر از RSA امنیتی معادل RSA را ارائه می‌دهد ضمن اینکه انتخاب پارامترها برای XTR در مقایسه با RSA خیلی سریع است.

هرچند که طول کلید ECC گاهی می‌تواند بیشتر از اندازهٔ کلید XTR کوچک شود، در بسیاری از کاربردها (مثلاً ذخیره‌سازی)، اندازهٔ کلید ECC و XTR قابل مقایسه خواهد بود. یک مزیت نسبی که XTR نسبت به خمهای بیضوی دارد این است که اساساً برپایهٔ همان مسألهٔ لگاریتم گسستهٔ مورد استفاده در DSS بنا شده است که به سریعتر پذیرفته شدن آن به عنوان یک الگوریتم قوی کمک می‌کند.

جدول ۴-۱۹، زمان اجرا و اندازهٔ کلید را برای RSA ، ECC و XTR مقایسه می‌کند.

جدول ۴-۱۹ اندازهٔ کلید RSA ، ECC و XTR و زمان اجرای RSA و XTR

	shared keysize	ID-based keysize	non-ID-based keysize	key selection	encrypting (verifying)	decrypting (signing)
1020-bit RSA	n/a	510 bits	1050 bits	1224 ms	5 ms	40 (no CRT: 123) ms
170-bit XTR	340	388 bits	680 bits	73 ms	23 ms	11 ms
170-bit ECC	171	304 bits	766 bits			

به‌طور خلاصه، XTR (طبق ادعای ایجادکنندگان آن) کارآ، فشرده و امن است و به سادگی پیاده‌سازی می‌شود.

۴-۷-۶-۳. کوله‌پشتی‌ها

تنها چند سیستم رمزنگاری کلید عمومی کوله‌پشتی مورد تمایل وجود دارند که البته هیچکدام اهمیت کاربردی ندارند.

◆ Rivest-Chor

این سیستم رمزنگاری برپایهٔ یک گونهٔ مشخص از مسألهٔ کوله‌پشتی بنا شده است و یکی از سیستم‌های رمزنگاری کوله‌پشتی بوده است که بهترین مقاومت را در برابر حملات داشته است.

[1] Trace representation
[2] Wireless Application Protocol/Wireless Transport Layer Security
[3] Internet Protocol SECurity / Internet Key Exchange
[4] Secure Electronic Transaction

- امنیت رمزکردن Chor-Rivest
  - هنگامی که پارامترهای سیستم به درستی (!) انتخاب شوند، حملهٔ ممکن شناخته شده‌ای برای آن وجود ندارد.
  - اگر قسمتهایی از کلید خصوصی فاش شود، سیستم ناامن می‌شود.
- پیاده سازی
  - رمزکردن، عمل خیلی سریعی می‌باشد؛ اما رمزگشایی، خیلی کند است.

یک اشکال بزرگ در روش Chor-Rivest این است که کلید عمومی بطور قابل‌توجهی بزرگ است (مثلاً ۳۶۰۰۰ بیت).

◆ Merkle-Hellman

بیشتر به دلایل تاریخی مهم است، چرا که اولین واقع‌سازی قوی یک روش رمزکردن کلید عمومی بود. بر ایدهٔ سادهٔ مخفی‌کردن مسألهٔ کوله‌پشتی "ماورای افزایشی"[1] ساده بوسیلهٔ پوشش[2] بنا شده بود. به هرحال، درسال ۱۹۸۰ شکسته شد. گونه‌های زیادی بعداً پیشنهاد شدند اما اغلب نشان داده شده است که ناامن می‌باشند.

۴-۶-۷-۴. شبکه‌ها(Lattices)

در سالهای اخیر تمایل زیادی به سمت سیستم‌های رمزنگاری بر پایهٔ شبکه ایجاد شده است. یکی از دلایل این امر این است که کلاسهای مشخصی از مسائل شبکه NP-hard هستند و چندین سیستم رمزنگاری کارآ نیز پیشنهاد شده است که قوی هم به‌نظر می‌رسند.

◆ NTRU

یک سیستم رمزنگاری است که در اواسط دههٔ ۱۹۹۰ به عنوان یک رمزکنندهٔ کلید عمومی کارآ ارائه شده است. این سیستم بر پایهٔ مسألهٔ Lattice بنا شده است و تعدادی خصوصیات جالب دارد.

بعضی از نسخه‌های اولیه مشکلاتی داشتند، اما نسخهٔ فعلی برای بعضی استانداردهای ایالات متحده پیشنهاد شده است. حتی الگوریتم امضای دیجیتالی برپایهٔ آن نیز پیشنهاد شده‌است (NSS[3]).

[1] Super increasing
[2] Masking
[3] NTRU Signature Scheme

۴-۷-۶-۵. "دیکد کد خطی"[1]

- **McEliece**

در سال ۱۹۷۸، برپایهٔ "کدهای تصحیح خطا"[2] و مسألهٔ دیکد کد خطی بنا شده است چراکه مسألهٔ دیکد یک کد خطی دلخواه، NP-hard می‌باشد. در این حالت، یک توصیف از کد اصلی به‌عنوان کلید خصوصی و یک توصیف از کد تغییر شکل یافته به‌عنوان کلید عمومی عمل می‌کند. McEliece وقتی‌که با کدهای Goppa استفاده شود در برابر رمزشکنی مقاوم است. این الگوریتم اولین رمزکردن کلید عمومی بوده است که از اتفاقی بودن (تصادفی) در فرآیند رمزکردن استفاده کرده است. با وجود کارایی بالا، درعمل بخاطر کلیدهای عمومی خیلی بزرگ میل زیادی برای استفاده از آن ایجاد نشده است.

۴-۷-۷. استانداردهای موجود یا در حال شکل‌گیری

استانداردها در چندین جهت ضروری هستند: روشهای رمزنگاری و ارائهٔ کلیدها. به علت اهمیت گروه الگوریتم‌های کلیدعمومی مجامع مختلفی استانداردسازی در بخش‌های مختلف آن را مورد توجه قرار داده‌اند. در این بخش از مهمترین فعالیت‌های انجام‌شده و یا در حال انجام نام برده می‌شود:

۴-۷-۷-۱. استانداردهای ANSI و ISO

استانداردهای ANSI و ISO نیاز به خریداری دارند و بنابراین لینک برای استانداردهای شخصی ندارند. آنها می‌توانند از طریق وب از ANSI و ISO خریداری شوند.

- *ANSI X9.30، X9.31 (برای امضاء RSA پیشنهاد شده است) ، X9.42، X9.44، X9.62 و X9.63*
- *ISO/IEC 9796، 10118، 14888*

۴-۷-۷-۲. NIST FIPS

NIST دو گروه الگوریتم‌های نامتقارن در FIPS 140-1/140-2 (که مربوط به استانداردهای رمزنگاری است) را در گروههای زیر تقسیم‌بندی می‌کند:

- *"استاندارد امضای دیجیتالی"[3] (FIPS 182-2)*

(ANSI X9.30) DSA ، (ANSI X.9.31) RSA ، (ANSI X9.62) ECDSA

[1] Linear code decoding
[2] Error-correcting codes
[3] Digital Signature Standard (DSS)

- *مدیریت کلید*

Diffie-Hellman (ANSI X9.42) ، RSA (ANSI X9.44) ، خمهای بیضوی (ANSI X9. 63) و Key Wrapping (پوشاندن کلید)

۴-۷-۷-۳. پروژهٔ NESSIE

در مورد این پروژه در بخشهای قبلی صحبت شده است. الگوریتم‌های فینالیست در قسمت رمزکردن کلید عمومی الگوریتم‌های زیر می‌باشند:

- *ACE Encrypt*
- *EPOC-2*
- *PSEC-2*
- *ECIES*
- *RSA-OAEP*

در مورد الگوریتم‌های نام برده شده در این بخش نمی‌توان قضاوت نمود و باید منتظر پایان پروژهٔ مربوطه بود.

۴-۷-۷-۴. PKCS

استانداردی است که توسط شرکت RSA Data Security ارائه شده‌است و اصطلاحاً یک استاندارد de facto می‌باشد؛ بدین معنی که در حقیقت وجود دارد اما یک استاندارد به‌صورت رسمی پذیرفته‌شده، نیست. در سندی که مربوط به وزارت دفاع آمریکا است این استاندارد (PKCS #1) در بخش استانداردهای امنیت اطلاعات به عنوان یک "استاندارد موجود" (Emerging Standard) به چشم می‌خورد (۲۱ ژوئن ۲۰۰۲).

۴-۷-۷-۵. پروژهٔ IEEE P1363

تکنیک‌های کلید عمومی زیادی پیشنهاد شده‌اند که هریک مزایای خود را دارد. به هرحال، یک مرجع مجرد وکامل معرف محدودهٔ کامل تکنیکهای کلید عمومی مشترک پوشش‌دهندهٔ Key agreement، رمزکردن کلید عمومی و امضای دیجیتالی، شناسایی[1] از چندین خانواده نظیر لگاریتم‌های گسسته، تجزیهٔ عدد صحیح و خم‌های بیضوی وجود ندارد. مقداری کار در هر زمینه انجام شده‌است، اما یک استاندارد مستقل کامل وجود ندارد.

هدف این پروژه تحمیل یک مجموعهٔ مشخص از تکنیک‌های کلید عمومی یا خصوصیات مشخصی از تکنیک‌های رمزنگاری نظیر اندازه‌های کلید، نمی‌باشد؛ بلکه هدف، مهیاکردن یک مرجع برای مشخصات تکنیک‌های گوناگونی است که درهر کاربردی ممکن است در نظر گرفته شود.

[1] Identification

مشخصات تکنیک‌های رمزنگاری کلید عمومی متداول مکمل آنهایی است که در IEEE P1363 در نظر گرفته می‌شود. درسال ۱۹۹۴ اولین جلسهٔ آن برگزار شد. بیشتر از ۲۰ جلسهٔ گروه کاری انجام شده‌است و درسال ۱۹۹۷، پروژه به دو قسمت P1363 و P1363a شکسته شد. بعداً این تقسیم‌بندی گسترش یافت و دو پروژهٔ P1363.1 و P1363.2 درنظرگرفته شد که هر کدام زمینه‌های مختلفی را شامل می‌شوند و بصورت موازی انجام می‌شوند. در اکتبر ۲۰۰۱، Submissionها بسته شد. در سال ۲۰۰۲ گروههایی هرسند را بررسی و مرور خواهند کرد و بالاخره در اواخر ۲۰۰۲ انتظار می‌رود نظر خود برای رأی‌گیری در مورد P1363.1 اعلام کنند و در اوایل ۲۰۰۳ نیز رأی‌گیری برای P1363.2 مورد انتظار است.

استاندارد IEEE P1363 به طور خلاصه شامل بخشها و مطالب زیر است:

- سه خانواده از توابع رمزنگاری، برحسب مسألهٔ مشکل از لحاظ محاسباتی زیربنایی
  - لگاریتم‌گسسته درگروه باقیمانده‌های به پیمانهٔ یک عدد اول (DL)
  - لگاریتم‌گسسته درگروه نقاطی واقع بر یک خم منحنی روی یک میدان متناهی (EC)
  - تجزیهٔ عدد صحیح (IF)
- برای خانوادهٔ DL، استاندارد شامل الگوریتم‌های زیر خواهد بود:
  - "توافق کلید"[1] Diffie-Hellman که به هر طرف اجازه می‌دهد تا دو جفت‌کلید داشته باشد.
  - توافق کلید Menezes-Qu-Vanstone که برای هرطرف دو جفت کلید نیاز دارد.
  - امضاهای DSA با توابع درهم‌سازی SHA-1 و RIPEMD-160
  - امضاهای Neyberg-Rueppel با پیوست SHA-1 و RIPEMD-160 به عنوان توابع درهم‌سازی
- برای خانوادهٔ EC، استاندارد خانوادهٔ DL منعکس (تکرار) خواهد شد؛ تنها فرق عمده تغییر درگروه زیربنایی آن می‌باشد.
- برای خانوادهٔ IF، استاندارد، شامل الگوریتم‌های زیر خواهد بود:
  - رمزکردن RSA با Padding متقارن بهینه (OAEP)[2]
  - امضای RSA با پیوست استفاده از یک تابع درهم‌سازی و Padding مقدارANSI X9.31 hash

[1] Key agreement
[2] Optimal Asymmetric Encryption Padding

- معادل‌های امضاهای RSA بالا توسط Rabin-Williams

- تعدادی از پیوستها به شرح زیر:

- پس‌زمینهٔ ریاضی رمزنگاری به همراه الگوریتم‌های نظریهٔ اعداد برای استفاده در پیاده‌سازی استاندارد
- معنی اجابت (پذیرفتن) استاندارد چیست؟
- توضیح اصول (اساس) تصمیمات گرفته‌شده در استاندارد
- ملاحظات امنیت برای الگوریتم‌های استاندارد
- فرمتهای پیشنهاد شده برای اهداف ریاضی و خروجی‌های الگوریتم و توضیح کار انجام شده

۴-۸. امضاهای دیجیتالی کلید عمومی

"عدم انکار"[1] برای بسیاری از کاربردهای تجاری حیاتی است چراکه راهی برای رفع مشاجرات بین طرفین برای اطمینان از یکدیگر مهیا می‌کند. در تجارت کلاسیک غیررمزنگاری، عدم انکار معمولاً با استفاده از امضاهای دست نوشته با قلم روی کاغذ تأمین می‌شود که "امضاهای فیزیکی" نامیده می‌شوند. امضاهای فیزیکی راهی را مهیا می‌کنند که بدین وسیله شخص را نسبت به گفته یا پیمانش متعهد می‌کند. باید توجه کرد که همیشه تشخیص تغییرات انجام شده در سند بعد از امضای آن بوسیلهٔ امضاهای فیزیکی جعل‌شده، ساده و ناچیز نیست. در واقع چنین مواردی گاهی اوقات، دلیلی برای منازعهٔ بین طرفین است و رفع آنها ممکن است نیازمند فرایندهای قانونی طویل و پرهزینه و خبره‌هایی در امضای فیزیکی و جعل، باشد. به‌علاوه، تکنیک‌های تشخیص تنها می‌توانند بر روی سند کاغذی اصلی امضاء شده اعمال شوند. کپی‌ها و نمابرها، به درستی عدم انکار را مهیا نمی‌کنند. در حقیقت، ایجاد کپی‌های تغییر داده‌شده به نحویکه حتی یک خبره هم قادر به تشخیص اصلی بودن آنها در مقایسه با کپی‌های تغییر نیافته نباشد، امکان‌پذیر است.

به هر حال، "امضاهای دیجیتالی کلیدعمومی"[2] یا به صورت خلاصه امضاهای دیجیتالی، اتوماسیون کامل فرآیندهای تصدیق و امضاء را مهیا می‌کنند. به‌علاوه، امضاهای دیجیتالی و سند امضاء شده به عنوان رشته‌هایی از بایت‌ها نمایش داده می‌شوند. بنابراین می‌توانند به سادگی کپی شوند، روی یک شبکه منتقل شوند و بایگانی شوند. این موارد مزیت‌های خیلی بزرگی به‌ویژه برای انجام تجارت از طریق اینترنت به‌همراه دارند. از طرفی امضاهای فیزیکی، واقعاً یک مشابه خوب برای امضاهای دیجیتالی کلید عمومی نیستند چراکه یک پیوند مستقیم بین آنها و سند امضاء شده وجود ندارد.

تفاوت رمزکردن و امضای دیجیتالی این است که رمزکردن، محرمانگی و کنترل دسترسی را حل می‌کند اما امضای دیجیتالی جامعیت داده، اعتبارسنجی و عدم انکار را برآورده می‌کند.

به‌طور خلاصه، یک امضای دیجیتالی یک مهر[3] می‌باشد که شخص روی داده می‌گذارد؛ منحصر به فرد است و به سختی جعل می‌شود؛ به‌علاوه امضاء اطمینان می‌دهد که هر تغییر انجام شده در داده‌ای که امضاء شده است بدون تشخیص نمی‌ماند.

۴-۸-۱. امضای دیجیتالی و کابردهای آن

تکنیک‌های امضای دیجیتالی کلید عمومی توانایی‌های جامعیت داده و اعتبارسنجی منبع را جهت افزایش قابلیت اعتماد داده در شبکه‌های کامپیوتری- جاییکه تمام مسیرهای ارتباطی نمی‌توانند به صورت فیزیکی محافظت شوند- مهیا می‌کنند (در توزیع نرم‌افزار، در کاربردهای پایگاه داده، جامعیت اطلاعات ذخیره شده در پایگاه داده غالباً حیاتی است). این فن‌آوری نخست در اوایل دههٔ ۸۰ با گسترش رمزنگاری کلید عمومی تعریف شد، اما اخیراً تمایل مجدد و تازه‌ای را به عنوان یک مکانیزم اعتبارسنجی در اینترنت به خود دیده است. به عنوان مثال، در زمان اتصال به یک سایت وب امن (HTTPS

[1] Non-repudiation
[2] Public Key Digital Signatures
[3] Stamp

URL)، برای خرید چیزی یک امضای دیجیتالی برای تأیید هویت سرور استفاه می‌شود. شبکه‌های بانکی که در یک شبکهٔ محلی گسترده توزیع شده‌اند می‌توانند مثال‌هایی اولیه برای آن باشند.

الگوریتم‌های پشتیبانی‌کنندهٔ امضاهای دیجیتالی کلید عمومی به‌صورت تاریخی مقادیر زیادی توان پردازشی را مصرف کرده‌اند، هرچندکه با پیشرفت‌هایی که اخیراً در پردازنده‌های مورد استفاده در PC ها و ایستگاه‌های کاری ایجاد شده است، دیگر در اغلب شرایط استفاده از آن، این مسأله یک نگرانی نمی‌باشد.

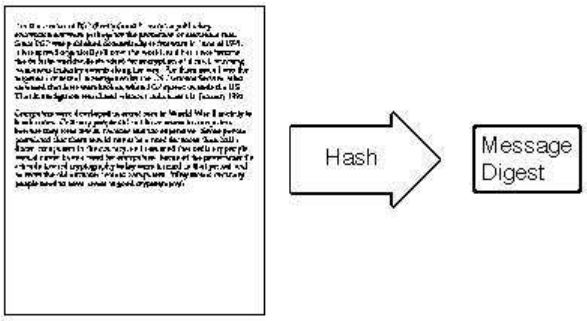
جامعیت داده و اعتبارسنجی می‌تواند با استفاده از رمزکردن کلید خصوصی و یک داور طرف سوم نیز مهیا شود. این راه، این عیب را دارد که طرف سوم باید مطمئن باشد و داده باید دو بار با دو کلید خصوصی جداگانه رمز و رمزگشایی شود. امضاهای دیجیتالی همچنین می‌توانند برای "مهر یا نشان زمان"[1] اسناد مورد استفاده قرار گیرند، یک طرف مورد اعتماد، سند و مهر زمان آن را با کلید محرمانهٔ خود امضاء می‌کند تا در زمان دیگری تصدیق کند که سند، در زمان بیان‌شده وجود داشته است. آنها همچنین می‌توانند برای "تصدیق" (یا گواهی دادن[2]) به اینکه یک کلید عمومی به یک شخص مشخص تعلق دارد، به کار روند. این عمل می‌تواند با امضای ترکیب کلید و اطلاعات در مورد صاحب آن به‌وسیلهٔ یک کلید مورد اعتماد انجام شود.

استفاده از یک کد اعتبارسنجی پیام و رمزنگاری کلید عمومی در فن‌آوری تکنیک‌های امضای دیجیتالی کلید عمومی ترکیب شده‌اند.

۴-۸-۲. نحوهٔ ایجاد و استفاده از امضای دیجیتالی

در ادامه، مراحل ایجاد یک امضای دیجیتالی با الگوریتم‌های کلید عمومی نشان داده می‌شود.

برای امضاء، یک نرم‌افزار، با فرآیندی به نام hashing (درهم‌سازی)، متن (داده) را به عبارت کوچکی، خلاصه می‌کند (شکل ۴-۳۷). واضح است که برگرداندن یک "خلاصه پیام" به دادهٔ اصلی که از آن ایجاد شده‌است، ممکن نیست.



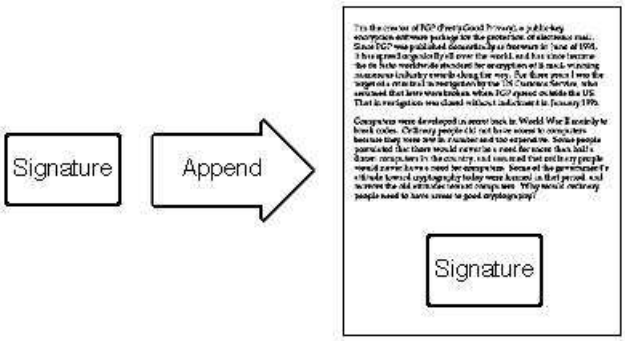
شکل ۴-۳۷ ایجاد خلاصه‌پیام از پیام اولیه

[1] Timestamp
[2] Certify

سپس، خلاصه‌پیام، با کلید خصوصی فرد امضاءکننده رمز می‌شود. حاصل امضای دیجیتالی می‌باشد (شکل ۴-۳۸).

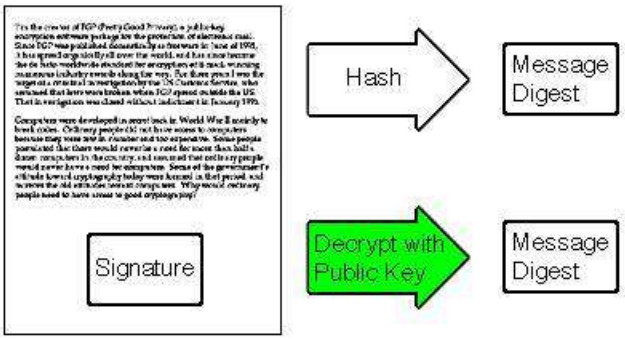
شکل ۴-۳۸ نحوهٔ تولید امضاء از خلاصه‌پیام

امضای دیجیتالی توسط نرم‌افزار، به سند، اضافه می‌شود (شکل ۴-۳۹). بدین‌ترتیب، همهٔ داده‌ای که hash شده بود، امضاء می‌شود.



شکل ۴- ۳۹ اضافه شدن امضاء به پیام

برای تأیید امضاء (شکل ۴-۴۰)، ابتدا نرم‌افزار گیرنده، امضاء را با استفاده از کلید عمومی فرستنده، رمزگشایی می‌کند. نتیجهٔ این عمل، به‌دست آوردن خلاصه‌پیامی است که در طرف فرستنده محاسبه شده است. همچنین نرم‌افزار مربوطه خود نیز خلاصه‌پیامی را از پیام دریافت شده محاسبه می‌کند.



شکل ۴- ۴۰ چگونگی تأیید یک امضاء

عمل لازم برای تأیید، مقایسهٔ این دو خلاصه‌پیام است. اگر این دو خلاصه‌پیام یکسان باشند، آن‌گاه ثابت می‌شود که فرستنده، سند را امضاء کرده است، زیرا فرض بر این است که فقط فرستنده، کلید خصوصی‌اش را دارد. الگوریتم مورد

استفاده برای تولید امضای دیجیتالی باید به نحوی باشد که بدون دانستن کلید محرمانه، ایجاد یک امضای معتبر، غیرممکن باشد.

۴-۸-۳. حملات ممکن علیه امضاهای دیجیتالی

علاوه بر مشکلات کلید خصوصی و توزیع و مدیریت کلیدهای عمومی حملاتی نیز محتمل می‌باشد. می‌توان کلاس‌های زیر را برای مدل‌های مختلف حمله در نظر گرفت:

- حملهٔ *Key-only*

در این حمله، دشمن تنها کلید عمومی امضاءکننده را می‌داند و بنابراین فقط توانایی بررسی صحت امضاهای پیام‌هایی را که به وی داده شده‌اند، دارد.

- حملهٔ *Known Signature*

دشمن، کلید عمومی امضاءکننده را می‌داند و جفت‌های پیام/امضاء که به‌وسیلهٔ صاحب امضاء انتخاب و تولید شده است را دیده است. این حمله در عمل امکان‌پذیر است و بنابراین هر روش امضایی باید در مقابل آن امن باشد.

- حملهٔ *Chosen Message*

به دشمن اجازه داده می‌شود که از امضاءکننده بخواهد که تعدادی از پیام‌های به انتخاب او را امضاء کند. انتخاب این پیام‌ها ممکن است به امضاهای از قبل گرفته شده بستگی داشته باشد. این حمله در غالب حالات، ممکن است غیرعملی به نظر برسد، اما با پیروی از قانون احتیاط، روش امضایی که در برابر آن ایمن است، ترجیح داده می‌شود.

- حملهٔ *Man-in-the-middle*

در این حمله، شخص از موقعیت استفاده کرده در هنگام مبادلهٔ کلید عمومی، کلید عمومی خود را جایگزین کرده و برای گیرنده می‌فرستد و بدین‌گونه می‌تواند به پیام‌ها دسترسی داشته باشد بدون اینکه فرستنده و گیرنده، مطلع باشند.

- *تعاریف مختلف برای یک حملهٔ موفقیت‌آمیز*
  - "جعل وجودی"[1]: دشمن در جعل امضای یک پیام- نه لزوماً به انتخابش- موفق می‌شود.
  - "جعل انتخابی"[2]: دشمن در جعل امضای بعضی پیام‌ها به انتخاب خودش موفق می‌شود.
  - "جعل سراسری"[3]: دشمن اگرچه‌که قادر به یافتن کلید محرمانه نیست، قادر به جعل امضای هر پیامی می‌باشد.
  - "شکستن کامل"[4]: دشمن می‌تواند کلید محرمانهٔ امضاءکننده را محاسبه کند.

[1] Existential Forgery
[2] Selective Forgery
[3] Universal Forgery
[4] Total Break

۴-۸-۴. الگوریتم‌های مورد استفاده برای امضای دیجیتالی

از نظر کاربرد، یک الگوریتم ds می‌تواند در نامه‌های الکترونیکی، انتقال پول (وجوه الکترونیکی)، مبادلهٔ دادهٔ الکترونیکی، توزیع نرم‌افزار و ذخیرهٔ داده و دیگر کاربردهایی که نیاز به اطمینان از جامعیت داده و اعتبارسنجی منبع داده دارد، استفاده شود. از نظر پیاده‌سازی، یک الگوریتم ds ممکن است در نرم‌افزار، firmware، سخت‌افزار یا هر ترکیبی از این‌ها پیاده‌سازی شود. غالب الگوریتم‌های مورد استفاده برای امضاهای دیجیتالی، الگوریتم‌های کلیدعمومی می‌باشند؛ هرچندکه در بعضی کاربردها از روشهای دیگری نیز استفاده می‌شود.

۴-۸-۴-۱. الگوریتم‌های کلیدعمومی

◆ RSA

الگوریتم امضای دیجیتالی RSA یک الگوریتم رمزنگاری پذیرفته شدهٔ FIPS برای تولید و تأیید امضای دیجیتالی می‌باشد که در ANSI X9.31 و PKCS #1 توضیح داده شده است.

رمزکردن و رمزگشایی RSA جابجایی‌پذیر هستند لذا می‌تواند مستقیماً به عنوان یک روش امضای دیجیتالی استفاده شود. فرض کنید یک مجموعه پارامترهای RSA یعنی {(e,n), (d,p,q)} داده شده‌اند. برای امضای یک پیام محاسبهٔ $S = M^d(\text{mod } n)$ و برای تأیید امضاء، محاسبهٔ $M = S^e(\text{mod } n) = M^{e.d}(\text{mod } n) = M(\text{mod } n)$ انجام می‌شوند. یعنی RSA می‌تواند به‌سادگی با معکوس‌کردن ترتیب نماهای استفاده شده برای رمزکردن و امضاء دیجیتالی استفاده شود. نمای خصوصی (d) برای ایجاد امضاء ونمای عمومی (e) برای تأیید امضاء استفاده می‌شود. البته معمولاً با استفاده از یک تابع درهم‌سازی ابتدا خلاصه‌پیام ایجاد می‌شود که همین خلاصه‌پیام نهایتاً امضاء می‌گردد.

قابل ذکر است که باید کلیدهای جداگانه‌ای برای اهداف امضاء و محرمانگی در هنگام استفاده از استاندارد X9.31 استفاده شوند چراکه الگوریتم RSA می‌تواند هم برای رمزکردن و هم امضای دیجیتالی استفاده شود.

◆ روش امضای ElGamal

رمزکردن ElGamal جابجایی‌پذیر نیست هرچندکه یک روش امضاء بسیار نزدیک و وابسته به آن وجود دارد. امنیت آن بستگی به مشکل بودن محاسبهٔ لگاریتم‌های گسسته دارد. مرحلهٔ setup (تولید کلید) یکسان است. با اشتراک عدد اول p و ریشهٔ اولیهٔ عمومی a هر کاربر یک عدد تصادفی را به عنوان (کلید) خصوصی انتخاب می‌کند و سپس کلید عمومی $y = a^x \text{ mod } p$ را محاسبه می‌کند. کلید عمومی (p، a و y) و کلید خصوصی (x) می‌باشد.

در عمل، برای امضاء، یک عدد تصادفی k انتخاب می‌شود که $GCD\ (k, p-1) = 1$ و سپس محاسبهٔ $K = a^k \text{ (mod } p)$ انجام می‌گردد. الگوریتم (معکوس) اقلیدسی بسط‌یافته برای یافتن S استفاده می‌شود.

$M = x.K + k.s \bmod (p-1)$,

$S = k^{-1} (M - x.K) \bmod (p-1)$.

امضاء، (K ،S) می‌باشد که k بعد از استفاده باید نابود شود. نظیر رمزکردن ElGamal، امضاء دو برابر اندازهٔ پیام می‌باشد. برای تأیید یک امضاء (S و K) روی پیام M باید اطمینان یافت که تساوی $y^K.K^S \bmod p = a^M \bmod p$ برقرار باشد.

◆ DSA (مورد استفاده در استاندارد DSS)

استاندارد DSS، به وسیلهٔ NIST و NSA دراوایل دههٔ ۹۰ ایجاد شد و یک گونه از الگوریتم‌های ElGamal و Schnorr می‌باشد. این استاندارد یک امضای ۳۲۰ بیتی با امنیت ۵۱۲ تا ۱۰۲۴ بیت ایجاد می‌کند. امنیت آن برپایهٔ مسألهٔ مشکل محاسبهٔ لگاریتم‌های گسسته می‌باشد و به‌طور گسترده پذیرفته شده است.

بعضی از خصوصیاتی که موجب گسترش استفاده از استاندارد DSS شده است می‌تواند موارد زیر باشد:

- استاندارد دولتی است.
- قابل استفاده برای رمزکردن نمی‌باشد.
- سرعت تولید امضاء، خوب است.

DSS محرمانگی را برای اطلاعات مهیا نمی‌کند. اگر محرمانگی لازم است، امضاءکننده باید نخست یک الگوریتم رمز کلیدمتقارن را به پیام اعمال کند و سپس آن را با استفاده از DSA امضاء کند.

- پارامترهای DSA عبارتند از:
  - p یک پیمانهٔ اول با شرط $2^{L-1} < p < 2^L$ برای ۱۰۲۴ => L => ۵۱۲ و L مضربی از ۶۴.
  - q یک مقسوم‌علیه اول از p-۱ که $2^{159} < q < 2^{160}$.
  - $g = h^{(p-1)/q} \bmod p$ که h هر عدد صحیح است با شرط ۱ < h < p-۱ به نحویکه $h^{(p-1)/q} \bmod p > 1$.
  - x = یک عدد صحیح تصادفی یا شبه‌تصادفی تولیدشده که در شرط 0 < x < q صدق می‌کند.
  - $y = g^x \bmod p$.
  - k = یک عدد صحیح تصادفی یا شبه‌تصادفی تولیدشده با شرط 0 < k < q.

اعداد صحیح p، q و g می‌توانند برای یک گروه از کاربران عمومی و مشترک باشند. یک جفت‌کلید خصوصی و عمومی یک کاربر به ترتیب x و y می‌باشد و به طور معمول برای یک دورهٔ زمانی ثابت هستند. پارامترهای x و k تنها برای تولید امضاء استفاده می‌شوند و باید محرمانه بمانند. پارامتر k برای هر امضاء باید دوباره تولید شود.

- ***تولید امضای DSA***

امضای پیام M، جفت اعداد r و s هستند که بر طبق معادلات زیر محاسبه می‌گردند:

$r = (g^k \bmod p) \bmod q$,

$s = (k^{-1} (SHA1(M) + xr)) \bmod q.$

که k^{-1}، وارون ضربی k به پیمانهٔ q می‌باشد، بدین معنی که $(k^{-1}k) \bmod q = 1$ و $0 < k^{-1} < q$. مقدار SHA-1(M) یک رشتهٔ خروجی ۱۶۰ بیتی به وسیلهٔ الگوریتم SHA-1 مشخص شده در FIPS 180-1می‌باشد. برای استفاده در محاسبهٔ s، رشته باید به یک عدد صحیح تبدیل شود.

به عنوان یک امر دلخواه، شخص باید بررسی کند که آیا r یا s صفر شده‌اند یا نه؟ اگر $r = 0$ یا $s = 0$ شد، یک مقدار جدید k باید تولید شود و امضاء باید دوباره محاسبه شود (اگر امضاء به طرز مناسبی تولید شود تا حد زیادی وقوع چنین امری غیرمحتمل است). امضاء به همراه پیام به "تأیید کننده"[1] (گیرنده) فرستاده می‌شود.

- ***تأیید امضای DSA***

قبل از تأیید امضاء در یک پیام امضاء شده، p، q و g به‌همراه کلید عمومی فرستنده برای تأییدکننده، به‌صورت معتبری باید در دسترس باشد.

فرض کنید M'، r' و s' به‌ترتیب، نسخه‌های دریافت‌شدهٔ M، r و s باشند و فرض کنید که y کلید عمومی امضاءکننده باشد. برای تأیید امضاء، تأییدکننده نخست چک می‌کند که آیا شرطهای $0 < r' < q$ و $0 < s' < q$ برقرارند یا نه. اگر یکی از شرایط نقض شود، امضاء، پذیرفته نمی‌شود.

اگر این دو شرط برآورده شوند، تأییدکننده محاسبات زیر را انجام می‌دهد:

$w = (s')^{-1} \bmod q,$

$u1 = ((SHA1(M')) w) \bmod q,$

$u2 = ((r') w) \bmod q,$

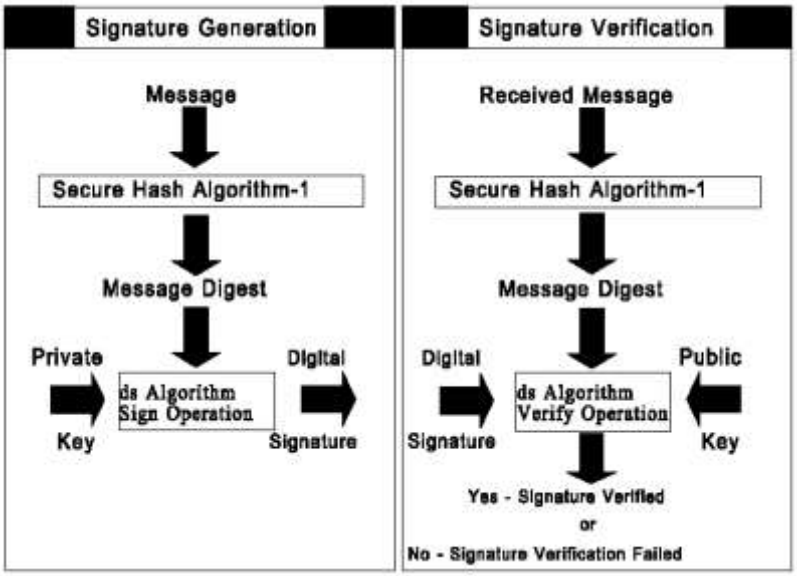
$v = (((g)^{u1} (y)^{u2}) \bmod p) \bmod q.$

[1] Verifier

اگر v = r'، آنگاه امضاء تأیید می‌شود و تأییدکننده می‌تواند اطمینان بالایی داشته باشد که پیام دریافت شده به وسیلهٔ طرفی فرستاده شده است که کلید محرمانهٔ x متناظر با y را داشته است.

اگر v مساوی r' نباشد، ممکن است پیام، تغییر داده شده باشد یا به‌صورت نادرستی توسط صاحب امضاء، امضاء شده باشد یا به‌وسیلهٔ یک دغل‌باز امضاء شده باشد و بنابراین پیام باید نامعتبر در نظر گرفته شود.

شکل ۴-۴۱ به طور خلاصه، نحوهٔ عمل DSA را نشان می‌دهد.



شکل ۴- ۴۱ نحوهٔ عمل DSA

بعضی از مشکلاتی که در مورد DSA می‌تواند ایجاد شود عبارتند از:

- خطر تغییرکردن داده‌های اولیه قبل از امضاءشدن (البته این خطر برای همهٔ روشهای امضاء موجود است). راه جلوگیری، امضاءکردن داده‌ها تا حدی که ممکن است نزدیک به زمان تولید می‌باشد (هدف در اینجا باید حداقل‌نمودن نقاط دسترسی دشمن یا دیگر منابع تداخل سیگنال باشد).
- اگر داده‌های دیجیتالی به‌سادگی امضاء شوند، بستهٔ دادهٔ امضاء شدهٔ حاصله نسبت به "حملهٔ تکرار"[1] آسیب‌پذیر است. در این سناریو، دشمن ترافیک پیام‌ها را که مناسب نیازش است، جمع می‌کند و در زمان مناسب، این ترافیک جمع‌شده را در جریان پیام تکرار می‌کند. راه جلوگیری، اضافه‌کردن تعدادی انواع نشانگر[2]های غیر تکراری در مجموعهٔ داده‌ای است که قرار است امضاء شود به نحوی‌که تکرار به آسانی به وسیلهٔ دریافت‌کننده تشخیص داده شود. به‌طور معمول Timestampها و "اعداد توالی"[3] بدین منظور استفاده می‌شوند.

[1] Replay attack
[2] Marker
[3] Sequence Number

یک سری تغییرات در FIPS 186-2 (استاندارد امضای دیجیتالی) در تاریخ ۵ اکتبر ۲۰۰۱ اعلام شد که در زیر به موارد تجدیدنظرشده اشاره می گردد:

- *اندازهٔ پیمانهٔ اول*

L می‌تواند تنها مقدار ۱۰۲۴ داشته باشد که در این صورت، پیمانهٔ P در محدودهٔ $2^{1023} < p < 2^{1024}$ خواهد بود. الگوریتم‌های RSA و Rabin-Williams استفاده‌شده در سیستم‌های موروثی با یک پیمانهٔ n و عوامل اول p، q از n تعریف می‌شوند. این تبصره مشخص می کند که n باید حداقل ۱۰۲۴ بیت باشد و p و q تقریباً n/۲ بیتی می‌باشند.

- *تولید اعداد تصادفی*

یک سری حملات دال بر عدم یکنواختی مولدهای اعداد شبه تصادفی (PRNGs)[1] علیه DSA اتفاق افتاده است که برای مقابله با آن‌ها باید تعداد امضاهای تولید شده با جفت کلید مشخص بیشتر از ۲ میلیون امضاء نباشد و یا این که PRNG ها رابابد اصلاح کرد.

◆ الگوریتم امضای دیجیتالی خم منحنی (ECDSA)

الگوریتم امضای دیجیتالی ECDSA، یک الگوریتم رمزنگاری پذیرفته شده برای تولید و تأیید امضای دیجیتالی می‌باشد. ECDSA مشابه خم منحنی DSA می‌باشد. ECDSA در ANSI X9.62 توضیح داده شده است. در این مورد باید ملاحظاتی در انتخاب طول کلید، میدان‌های زیرلایه، پایه (basis) خمها و"نقاط پایه"[2] صورت گیرد.

◆ Fiat-Shamir

این الگوریتم اساساً یک روش امضاء برپایهٔ مشکل یافتن ریشه‌های دوم به پیمانه pq است و دارای حق امتیاز و پیشروی DSA می‌باشد.

◆ Schnorr

این الگوریتم ایده‌های ElGamal و Fiat-Shamir را ترکیب می کند و از توان‌رسانی به پیمانهٔ p و پیمانهٔ q استفاده می کند. غالب محاسبات می‌تواند در یک فاز پیش محاسبه قبل از امضاء کامل شود. برای یک سطح امنیت، امضاها به طرز قابل توجهی کوچکتر از RSA می‌باشد و دارای حق امتیاز است.

۴-۸-۴-۲. توابع درهم‌سازی کلیددار

همهٔ روش‌های امضای بالا، از الگوریتم‌های کلید عمومی استفاده می کنند ولی با توجه به سربارهای هزینه و طولی که آنها ایجاد می کنند، نیاز برای یک روش امضای کلید خصوصی پیدا می‌شود. پیشنهاد جالب استفاده از توابع درهم‌سازی سریع بود مشتمل بر یک کلید به همراه پیام بود.

[1] Pseudo-Random Number Generators
[2] Base Points

KeyedHash = Hash (Key | Message).

ضعفهایی در پیشنهاد اصلی موجود بود که باعث پیشنهادی نظیر زیر شد.

KeyedHash = Hash (Key1 | Hash (Key2 | Message)).

HMAC نتیجهٔ ایدهٔ استفاده از تابع درهم‌سازی کلیددار بود. به عنوان یک استاندارد اینترنت (RFC2104) مشخص شد. این روش از تابع درهم‌سازی روی پیام به شکل زیر استفاده می‌کند.

$HMAC_K$ = Hash ((K+ XOR opad) | | Hash ((K+ XOR ipad)| |M)).

که K+ خروجی pad شده می‌باشد و opad و ipad مقادیر Padding مشخص شده می‌باشند. سربار این عمل تنها ۳ تا محاسبهٔ hash بیشتر از نیاز پیام می‌باشد. امنیت آن مستقیماً به امنیت تابع درهم‌سازی زیرلایهٔ آن وابسته است. برای این منظور هر کدام از الگوریتم‌های hash MD5، SHA-1 و RIPEMD-160 می‌توانند استفاده شوند.

۴-۸-۵. استانداردهای موجود یا در حال شکل‌گیری

۴-۸-۵-۱. پروژهٔ NESSIE

الگوریتم‌های نهایی در بخش امضاهای دیجیتالی عبارتند از:

- ECDSA : Certicom Corp., USA and Certicom Corp., Canada
- ESIGN : Nippon Telegraph and Telephone Corp., Japan
- RSA-PSS : RSA Laboratories Europe, Sweden and RSA Laboratories, USA
- SFLASH : BULL CP8, France
- QUARTZ : BULL CP8, France

به ادعای ایجادکنندگان‌شان، این الگوریتم‌ها، الگوریتم‌های کارا، سریع، مؤثر و امنی می‌باشند. به نظر می‌رسد باید تا پایان پروژه (اواخر سال ۲۰۰۲) منتظر ماند تا تمام تحلیل‌های ممکن صورت پذیرد.

۴-۸-۵-۲. روشهای مبتنی بر الگوریتم‌های جدید

از این دسته از روشها می‌توان از NSS (NTRU Signature Scheme) نام برد که برپایهٔ الگوریتم جدید NTRU بنا شده است.

۴-۸-۵-۳. FIPS PUB 186-2 (۲۷ ژانویهٔ ۲۰۰۰)

در FIPS 186-2 سه الگوریتم معرفی شده‌اند: (برای همهٔ این الگوریتم‌ها، استفاده از تابع درهم‌سازی SHA-1 توصیه شده‌است)

- DSA (ANSI X9.30)
- RSA (ANSI X9.31) که در دورهٔ انتقال (از ژولای ۲۰۰۱ تا دسامبر ۲۰۰۲) از PKCS#1 (نسخهٔ ۱٫۵ یا بالاتر) استفاده می‌شود (این نوع RSA، گاهی اوقات rDSA نامیده می‌شود).
- ECDSA (ANSI 9.62)

با در نظرگرفتن پاره‌ای از ملاحظات نیاز به ایجاد اصلاحاتی در این استاندارد دیده می‌شود. از آنجمله ارائهٔ استانداردهای جدید رمزنگاری (نظیر AES) می‌باشد. DSA هم اکنون به ۱۰۲۴ بیت محدود است. درحالیکه ۱۲۸ بیت AES حدوداً معادل ۳۰۰۰ بیت DSA می‌باشد. از طرفی، ۱۰۲۴ بیت DSA حدوداً به استحکام ۱۶۰ بیت SHA-1 می‌باشد. با درنظرگرفتن SHA256، SHA384 و SHA512 (استانداردهای جدید SHA) واضح است که نیاز به بسط اندازهٔ کلیدها احساس می‌شود.

همچنین هنوز تردید در اجازه دادن به استفاده از(PKCS#1(RSA وجود دارد؛ لذا از طرحهای آیندهٔ DSS، اصلاح از FIPS 186-2 به FIPS 186-3 می‌باشد که پیش‌نویس آن در فوریهٔ ۲۰۰۱ در دسترس قرار گرفته است.

۴-۹. جمع‌بندی

نیازهای امروزی ارتباطی بشر برای اهداف مختلف نیازمند ملزومات خاص هر ارتباط نیز می‌باشد. به عنوان مثال ایمن بودن ارتباطات ازجملهٔ مهمترین خصوصیات موردنظر در بسیاری از زمینه‌ها می‌باشد. الگوریتم‌های رمزنگاری نقش عمده‌ای در فراهم‌نمودن بستری امن برای ارتباطات مختلف بر عهده دارند. در واقع از الگوریتم‌های رمزنگاری در بسیاری از قراردادهای مهم استفاده می‌شود که در مطالب فصل به بعضی از آنها اشاره شده است.

انتخاب سیستم رمزنگاری به نوع کاربرد و اهم پارامترهای مورد نظر (مانند امنیت، کارآیی، استانداردها و قابلیت کارمشترک) بستگی دارد. در انتخاب یک الگوریتم رمز مناسب برای یک کاربرد باید با رمزشکنی و حملات علیه سیستم‌های رمزنگاری آشنا بود و الگوریتمی باید انتخاب شود که در برابر حملات شناخته شده مقاوم باشد. هر چند که قدرت الگوریتم از اهمیت بیشتری نسبت به طول کلید (در الگوریتم‌هایی که از کلید استفاده می‌کنند) برخوردار است، اما طول کلید نیز عامل مهمی در بالابردن امنیت الگوریتم می‌باشد. همچنین ملاحظات زیادی در مورد مدیریت کلید باید صورت گیرد.

با توجه به ساختار این فصل نتیجه‌گیریهای زیر از بخش‌های مختلف آن قابل حصول است. این نتیجه‌گیری‌ها به تفکیک گروه‌های مختلف الگوریتم‌های رمزنگاری مورد بحث بیان شده‌اند.

۴-۹-۱. رمزکننده‌های بلوکی (متقارن)

ازجملهٔ الگوریتم‌های خیلی معروف در زمینهٔ رمزنگاری الگوریتم DES می‌باشد. الگوریتم DES برای استفاده در حدود یک دهه طراحی شده بود اما طراحی خوب آن باعث شد که حدود ۲۵ سال مورد استفاده قرارگیرد. بنابراین پرداختن به آن نه‌تنها امری اضافی نیست بلکه در فهم خیلی از مفاهیم مربوط به الگوریتم‌های متقارن ضروری می‌باشد. البته NIST در سال ۱۹۹۹ Triple-DES را به عنوان الگوریتم جانشین معرفی کرد و استفاده از Single-DES را تنها در سیستم‌های موروثی به‌جا مانده از نسل قدیم اجازه داد. خریدهای جدید برای پشتیبانی سیستم‌های موروثی، در مواردی که امکان دارد، باید از محصولات Triple-DES در پیکربندی Single-DES استفاده کنند.

همان‌طورکه در متن گزارش نیز آمده‌است DES و به‌ویژه Triple-DES همچنان در پروتکل‌های معروفی نظیر SSL[1]، PCT[2]، PGP[3]، SGC[4]، SET[5]، TLS[6] مورد استفاده قرار می‌گیرند ولی بتدریج جای خود را به الگوریتم‌های بهتر (AES) خواهند داد.

AES استاندارد جدید رمزنگاری متقارن زیر نظر NIST می‌باشد. انتظار می‌رود که AES امنیت رمزنگاری قوی را برای حفاظت اطلاعات حساس در قرن بیست‌ویکم فراهم کند. الگوریتم رمزنگاری AES همانطور که در بخشهای قبلی گفته شد به دلیل استحکام و انعطاف‌پذیری خوب به عنوان استانداردی در بین الگوریتم‌های متقارن بلوکی معرفی شده است. این در حالی است که چگونگی انتخاب آن کاملاً مشخص و واضح بوده‌است. به نظر می‌رسد که AES با مقبولیت زیادی روبرو شود. به عنوان مثال در طراحی بسیاری از الگوریتمهای جدید رمزنگاری طراحان آنها نگاه ویژه‌ای به نحوهٔ طراحی AES داشته‌اند و حتی‌الامکان از آن استفاده نموده‌اند.

۴-۹-۲. رمزکننده‌های جریانی

به دلیل خصوصیات ویژهٔ رمزکننده‌های جریانی استفادهٔ از آنها وابسته به شرایط و خصوصیات کاربرد آن می‌باشد. به عنوان مثال در صورت مواجه‌بودن با حجم کثیری از داده یا نیاز به بلادرنگی و یا کمبود بافر استفاده از آنها توصیه می‌شود منتها باید توجه داشت که سطح امنیت کاهش پیدا می‌کند. بنابراین اگر امنیت مهم‌ترین پارامتر باشد توصیه می‌شود که از این گروه از رمزکننده‌ها استفاده نشود (البته امنیت بعضی از الگوریتم‌های جدید مطرح‌شده در این گروه هنوز به طور کامل مشخص نیست).

[1] Secure Socket Layer
[2] Private Communication Technology
[3] Pretty Good Privacy
[4] Server Gated Cryptography
[5] Secure Electronic Transactions
[6] Transport Layer Security

الگوریتم‌های متداول پیش از این در این گروه RC4 و SEAL بوده‌اند که گهگاه مشکلاتی در آنها گزارش شده‌است (شکسته‌شدن RC4 ۴۰ بیتی)؛ به همین دلیل الگوریتم‌های جایگزینی پیشنهاد شده‌اند (مثلاً Scream به جای SEAL).

RC4 در پروتکل‌هایی نظیر PCT[1]، SGC[2] (توسعهٔ مایکروسافت به SSL از RC4 ۱۲۸ بیتی استفاده می‌کند)، SET[3]، SSL (SSL v2 از RC4 ۴۰ بیتی و SSL v3 از RC4 ۱۲۸ بیتی استفاده می‌کند) مورد استفاده قرار گرفته است.

به طور کلی باید چشم به پروژه‌هایی نظیر NESSIE دوخت تا سرانجام شاهد وجود استانداردهایی برای این گروه از رمزکننده‌ها باشیم.

۴-۹-۳. توابع درهم‌سازی

خوشبختانه در گروه الگوریتم‌های درهم‌سازی الگوریتم‌های استاندارد شده و یا در حال استاندارد شدن وجود دارند که نگرانی استفاده از آنها را رفع می‌کند. به‌طور مشخص سری الگوریتم‌های SHA (SHA-1، SHA-256، SHA-384 و SHA-512) فعلاً گزینهٔ خوبی به نظر می‌رسد در حالیکه الگوریتم‌های خوب دیگری نظیر RIPEMD-160 ، Whirlpool و Tiger (بهینه‌شده برای پردازنده‌های ۶۴ بیتی) نیز موجودند.

همانطورکه در متن گزارش نیز آمده‌است؛ MD5 هنوز هم در کاربردهایی استفاده می‌شود اما به‌هیچ وجه در کاربردهایی که نیاز به مقاومت در برابر تصادم دارند توصیه نمی‌شود.

۴-۹-۴. الگوریتم‌های رمزنگاری کلیدعمومی

روشهای رمزکردن کلید عمومی ذاتاً کندتر از الگوریتم‌های رمزکردن کلید متقارن می‌باشند. به این دلیل، درعمل معمولاً برای نقل و انتقال کلیدهایی که بعداً برای رمزکردن تودهٔ داده با الگوریتم‌های متقارن و سایر کاربردهایی که شامل جامعیت داده و اعتبارسنجی می‌باشند و برای رمزکردن آیتمهای دادهٔ کوچک نظیر شماره‌های کارتهای اعتباری و PIN ها استفاده می‌شوند.

تکنیک‌های RSA، Diffie-Hellman و ElGamal بیتهای بیشتری برای کلیدها برای یک امنیت معادل در مقایسه با الگوریتم‌های متقارن نوعی استفاده می‌کنند. یک کلید ۱۰۲۴ بیتی دراین سیستم‌ها حدوداً معادل یک کلید متقارن ۸۰ بیتی می‌باشد و به نظر خیلی‌ها این تعداد حداقل طولی است که امروزه باید مورد استفاده قرارگیرد. اگرکه نیاز به بیتهای کمتری برای یک سیستم کلید عمومی باشد، باید از رمزنگاری خم منحنی استفاده شود (یافتن خمهای مناسب مشکل است، پروژهٔ IEEE P1363 تعدادی خم را پیشنهاد داده است). به هرحال باید مراقب بود، زیرا رمزنگاری خم منحنی دارای حق امتیاز

[1] Private Communication Technology
[2] Server Gated Cryptography
[3] Secure Electronic Transaction

نیست، اما تکنیک‌های مشخصی برای تسریع آن دارای حق امتیاز هستند (خم منحنی در استفادهٔ معمول آن برای رمزکردن کلیدهای رمزجلسه/توده به اندازهٔ کافی سریع است که واقعاً نیاز به این تسریع‌ها نداشته باشد).

با این همه، ECC مشخصاً درکاربردهایی که حافظه، پهنای باند و/یا توان محاسباتی، محدود است (نظیر یک کارت هوشمند) مفید است و در این زمینه است که انتظار می‌رود استفاده از ECC گسترش یابد. البته نباید از الگوریتم XTR نیز غافل شد که در حال گسترده‌شدن می‌باشد و یک رقیب جدی برای خمهای بیضوی محسوب می‌گردد.

در گروه الگوریتم‌های بر پایهٔ شبکه نیز NTRU شهرت خوبی کسب کرده است که به علت خصوصیات خوب و استحکام مناسب آن می‌باشد. NTRU و امضای بر پایهٔ آن (NSS) در حال واردشدن در بدنهٔ بعضی استانداردها است و احتمالاً بر محبوبیت آن نیز اضافه خواهد شد.

۴-۹-۵. الگوریتم‌های امضای دیجیتالی

پیاده‌سازی تکنیک‌های امضای دیجیتالی کلید عمومی نیازمندی‌هایی را روی شبکه به‌وجود می‌آورد. همان توابع خلاصه‌پیام و الگوریتم کلید عمومی که برای ایجاد امضای دیجیتالی استفاده شده‌اند باید توسط فرستنده و گیرنده استفاده شوند. جفت‌کلیدهای عمومی/خصوصی باید تولید و و نگهداری شوند. کلیدهای عمومی باید توزیع شوند (یا در محلی که در دسترس عموم باشند، قرار گیرند) و کلیدهای خصوصی باید محافظت شوند.

لذا، استفاده از این فن‌آوری نیاز به مدیریت شبکهٔ کارکنان با دانش رمزنگاری کلید عمومی و استفاده از نرم‌افزاری که الگوریتم‌های رمزنگاری کلید عمومی و امضای دیجیتالی را پیاده‌سازی می‌کند، همچنین امنیت کارکنان و نرم‌افزاری که می‌تواند کلیدهای رمزکردن/رمزگشایی را تولید، توزیع و کنترل کند و پاسخگوی فقدان و مصالحه (افشای) کلید باشد، دارد.

پیشنهاد خوب برای استفاده از الگوریتم‌های امضای دیجیتالی همانطور که در خلال گزارش بیان شده است DSS (آخرین نسخه که شامل RSA، DSA و ECDSA می‌باشد) و روشهای جدید نظیر NSS می‌باشند. البته پروژهٔ NESSIE نیز که در حال اتمام است الگوریتم‌های خوبی را پیشنهاد خواهد داد. این درحالی است که تمایلات اخیر به سمت اندازه‌های امضای خیلی کوتاه می‌باشد.

۴-۱۰. فهرست منابع

[1] Cryptographic Protocols and Standards, http://www.ssh.com/support/cryptography/index.htmlcrypto/protocols.cfm.

[2] Adam Shostack , An Overview of SHTTP, May 1995.

[3] E. Rescorla, A. Schiffman, The Secure HyperText Transfer Protocol, RFC 2660, August 1999.

[4] Burton S. Kaliski Jr, An Overview of the PKCS Standard, An RSA Laboratory Technical Note.

[5] SSH Protocol Overview, http://www-comnet.technion.ac.il/~cn19s01/ssh_overview.pdf.

[6] Niels Ferguson, Bruce Schneier, A Cryptographic Evaluation of IPsec, http://www.counterpane.com/ipsec.html, 2002.

[7] Arjen K. Lenstra, Eric R. Verheul, Selecting Cryptographic Key Size, http://www.simovits.com/archive/cryptosizes.pdf,and http://www.cacr.math.uwaterloo.ca/conferences/1999/ecc99/lenstra.doc, 24 November 1999.

[8] Morris Dworkin, Recommendation for Block Cipher Modes of Operation, NIST Special Publication 800-38A, 2001 Edition.

[9] Cryptographic Toolkit (Modes of Operation), http://csrc.nist.gov/encryption/tkmodes.html, CSRC, 2001.

[10] Modes of Operation, http://csrc.nist.gov/encryption/modes/, CSRC, 2002.

[11] SSH - Tech Corner – Introduction to Cryptography, http://www.ssh.fi/tech/crypto/intro.cfm, SSH Communications Security, 2002.

[12] Terry Ritter, Ritter's Crypto Glossary and Dictionary of Technical Cryptography, http://www.ciphersbyritter.com/GLOSSARY.HTM, 2002.

[13] Gary McGraw and John Viega, developerWorks: Security: Make your software behave: Everything to hide, http://www-106.ibm.com/developerworks/security/library/everything.html?dwzone=security, IBM, May 2000.

[14] Gary C. Kessler, An Overview of Cryptography, http://www.garykessler.net/library/crypto.html, 9 June 2002.

[15] M. Atreya, http://download.cryptoform.com/rsacryptography.pdf.

[16] FIPS 46-3, Data Encryption Standard (DES), US NIST, Reaffirmed 1999 October 25, http://csrc.nist.gov/publications/fips/.

[17] Anuj Seth, Data Encryption Page, http://www.anujseth.com/crypto/, 2002.

[18] H. Fruehauf, Zyfer Encryption Fundamentals, October 2001, Http://www.zyfer.com/research/briefings/pdf/Encryption-Fund'tls_9-01.pdf.

[19] Cryptographic Toolkit, http://csrc.nist.gov/encryption/.

[20] RSA Security FAQs, http://www.rsasecurity.com/rsalabs/faq/, 2002.

[21] http://www.faqs.org/faqs/cryptography-faq.

[22] B. Schneier, Applied Cryptography, New York: John Wiley & Sons, Second Edition, 1996.

[23] W. Stallings, Cryptography and Network Security, New Jersey: Prentice-Hall, Second Edition, 1999.

[24] Cryptography A-2-Z, http://www.ssh.fi/tech/crypto/, SSH Communications Security, 2002.

[25] Cryptography Archives, http://kremlinencrypt.com/crypto/, KremlinEncrypt.com.

[26] David A. Wheeler, Cryptographic Algorithms and Protocols, http://www.tldp.org/HOWTO/Secure-Programs-HOWTO/crypto.html, 2002.

[27] Deciphering Cryptography, http://www.ntsecurity.net/Articles/Index.cfm, Penton Media, Inc, 2002.

[28] Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997. (Free at www.cacr.math.uwaterloo.ca/hac)

[29] AES A Crypto Algorithm for the Twenty-first Century, http://aes.nist.gov/default.htm, November 2001.

[30] AES Home Page, Advanced Encryption Algorithm, http://csrc.nist.gov/encryption/aes/, December 2001.

[31] FIPS 197, Announcing the ADVANCED ENCRYPTION STANDARD (AES), November 2001.

[32] Report on the Development of the Advanced Encryption Standard (AES), October 2000.

[33] The AES Candidates, http://www.ddj.com/documents/s=913/ddj9812b/9812bs3.htm.

[34] Lars R. Knudsen and Vincent Rijmen, The Block Cipher Lounge – AES, http://www.ii.uib.no/~larsr/aes.html, 1999.

[35] Paulo Barreto's Crypto Page, http://planeta.terra.com.br/informatica/paulobarreto/, 2002.07.17.

[36] Standards Cryptographic Algorithms, http://www.wiltsec.co.uk/standards/algorithms.htm, December 2001.

[37] Lars R. Knudsen and Vincent Rijmen, The Block Cipher Lounge, http://www.ii.uib.no/~larsr/bc.html, 1999.

[38] Wei Dai, Cryptographic Algorithms, http://www.eskimo.com/~weidai/algorithms.html, 27 June 2000.

[39] Dr. Lawrie Brown, Cryptography - Lecture 12 - Modern Stream Ciphers, http://www.cs.adfa.edu.au/teaching/studinfo/ccs3/lectures/less12.html, 7 Nov. 2001.

[40] Cryptology ePrint Archive, http://eprint.iacr.org/2002/019/, 5 June 2002.

[41] Lauri Pesonen, GSM Interception, http://www.dia.unisa.it/ads.dir/corso-security/www/CORSO-9900/a5/Netsec/netsec.html, 21 Dec. 1999.

[42] IST Project Fact Sheet- NESSIE, http://dbs.cordis.lu/fep-cgi/srchidadb?ACTION=D&SESSION=107562002-8-17&DOC=26&TBL=EN_PROJ&RCN=EP_RCN_A:54113&CALLER=PROJ_IST

[43] John J. G. Savard, Stream Ciphers, http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/co0411.htm, 2000.

[44] S. Halevi, D. Coppersmith, C. Jutla, Scream: a software-e client stream cipher, IBM T. J. Watson Research Center, 5 June 2002.

[45] P. Ekdahl, T. Johansson, SNOW - a new stream cipher, Dept. of Information Technology Lund University: Sweden, http://www.it.lth.se/cryptology/snow/, 22 Nov. 2001.

[46] What is a Stream Cipher, http://www.hack.gr/users/dij/crypto/overview/streamciphers.html

[47] Stream Ciphers, http://www.tcs.hut.fi/~helger/crypto/link/stream/, 9 June 2002.

[48] Rick Wash, Lecture Notes on Stream Ciphers and RC4.

[49] P. Rogaway, D. Coppersmith, A Software_Optimized Encryption Algorithm: SEAL, 5 Sep. 1997.

[50] G. Rose, P. Hawkes, The t-Class of SOBER Stream Ciphers, QUALCOMM Australia, DRAFT: October 12, 1999.

[51] Bart Preneel, NESSIE PROJECT ANNOUNCES SELECTION OF CRYPTO ALGORITHMS, http://www.cryptonessie.org/, 2001.

[52] http://cnscenter.future.co.kr/crypto/algorithm/block.html.

[53] Private Key Cryptography, http://www.anujseth.com/crypto/privatekey.html.

[54] I broke Hal's SSL challenge, http://pauillac.inria.fr/~doligez/ssl/.

[55] Paulo S. L. M. Barreto, The Hashing Function Lounge, http://planeta.terra.com.br/informatica/paulobarreto/hflounge.html, 2002.08.31.

[56] John J. G. Savard, One-way Hash Functions, http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/mi0605.htm,

[57] John J. G. Savard, Description of SHA-1 and SHA-256, http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/mi060501.htm.

[58] Anuj Seth, hash Functions, http://www.anujseth.com/crypto/hash.html, August 2002.

[59] David Hopwood, MessageDigest algorithms, http://www.users.zetnet.co.uk/hopwood/crypto/scan/md.html, 2001.

[60] Antoon Bosselaers, The hash function RIPEMD-160, http://www.esat.kuleuven.ac.be/~bosselae/ripemd160.html, August 1999.

[61] Secure Hash Standard (SHS), http://csrc.nist.gov/cryptval/shs.html, NIST, April 2001.

[62] CRYPTOGRAPHIC TOOLKIT-Secure Hashing, http://csrc.nist.gov/encryption/tkhash.html, CSRC, FIPS 180-2, August 2002.

[63] R. Anderson, E. Biham, Tiger: A Fast New Hash Function, Cambridge Univ. and Technion, http://www.cs.technion.ac.il/~biham/Reports/Tiger/tiger/tiger.html.

[64] Paulo S. L. M. Barreto, V. Rijmen, The Whirlpool Hashing Function, http://planeta.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html, 2001.09.25.

[65] Cryptographic Algorithms, http://kremlinencrypt.com/crypto/algorithms.html.

[66] Cryptographic Algorithms, http://www.cl.cam.ac.uk/Research/Security/studies/st-alg.html, February 1999.

[67] Cryptography FAQ (07/10: Digital Signatures), http://www.faqs.org/faqs/cryptography-faq/part07/.

[68] M.J.B. Robshaw, On Recent Results for MD2, MD4, and MD5, RSA Data Security, 1996.

[69] SSH - Tech Corner – Cryptographic Algorithms, http://www.ssh.fi/tech/crypto/algorithms.cfm, SSH Communications Security, 2002.

[70] An Introduction to Computer Security: The NIST Handbook, NIST, Special Publication 800-12, http://csrc.nist.gov/nistpubs/800-12/handbook.pdf, chapt. 19.

[71] Elaine Barker, NIST Cryptographic Standards Toolkit, http://csrc.nist.gov/encryption/, 2001.

[72] Mohan Atreya, Introduction to Cryptography, http://www.rsasecurity.com/solutions/developers/whitepapers/IntroToCrypto.pdf.

[73] Joint Technical Architecture List of Mandated and Emerging Standards, Department of Defense, http://www.disa.mil/, 21 June 2002.

[74] Paulo Sérgio L. M. Barreto, Paulo Barreto's cryptography page, http://planeta.terra.com.br/informatica/paulobarreto/, 2002.

[75] Michael Liu, http://www.usc.edu/dept/engineering/illumin/vol3issue2/crypto/index1.html, 2002.

[76] John J. G. Savard, A Cryptographic compendium, http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/jscrypt.htm, 1999.

[77] IEEE P1363: Standard Specifications For Public Key Cryptography, http://grouper.ieee.org/groups/1363/P1363/, 2000.

[78] Lawrie.Brown, Cryptography - Lecture 16 - Public Key Encryption Algorithms, http://www.kewlstuff.co.za/cryptolessons/Lecture%2016.htm or http://www.cs.adfa.edu.au/teaching/studinfo/ccs3/lectures/less16.html, 2001.

[79] J. Lopez, R. Dahab, An Overview Of Elliptic Curve Cryptography, 2000.

[80] Cryptography Resources, http://www.certicom.com/resources/index.html, 2002.

[81] The XTR Public Key Cryptosystem, http://www.ecstr.com/, 2002.

[82] Public Key Digital Signatures, http://www.sei.cmu.edu/str/descriptions/pkds_body.html, Carnegie Mellon University, September 2000.

[83] Digital Signature, http://www.cscap.nuctrans.org/Nuc_Trans/links/dsa-disa.html, 2002.

[84] Announcing FIPS 186-2, Digital Signature Standard (DSS), http://cs-www.ncsl.nist.gov/encryption/dss/fr000215.html, DEPARTMENT OF COMMERCE – NIST, 2001.

[85] Cryptographic Toolkit (Digital Signatures), http://csrc.nist.gov/encryption/tkdigsigs.html, NIST-CSRC, 2002.

[86] DIGITAL SIGNATURE STANDARD, http://www.itl.nist.gov/lab/bulletns/archives/csl94-11.txt, NIST.

[87] Chapter 6. Public Key Cryptography, http://www.cs.tau.ac.il/~ah/Notes/n4.pdf.

[88] Matthew Meyer, How Java, cryptography, and law affect application developers, http://www.serverworldmagazine.com/sunserver/2000/09/javalaw.shtml, Publications & Communications Inc. (PCI), 2000.

[89] David Youd, What is a Digital Signature? http://www.youdzone.com/signature.html, 1996.

[90] Lawrie.Brown, Cryptography - Lecture 19 - Digital Signature Algorithms, http://www.kewlstuff.co.za/cryptolessons/Lecture%2016.htm or http://www.cs.adfa.edu.au/teaching/studinfo/ccs3/lectures/less16.html, 2001.