Cryptography for network security: failures, successes and challenges

Bart Preneel

Institutions: Katholieke Universiteit Leuven

Published on: 08 Sep 2010 - Mathematical Methods, Models, and Architectures for Network Security Systems

Topics: Cryptographic primitive, Cryptographic protocol, Security of cryptographic hash functions, Key management and Network security

Cryptography for Network Security: Failures, Successes and Challenges

Bart Preneel

Katholieke Universiteit Leuven and IBBT
Dept. Electrical Engineering-ESAT/COSIC,
Kasteelpark Arenberg 10 Bus 2446, B-3001 Leuven, Belgium
bart.preneel@esat.kuleuven.be

Abstract. This article discusses the state of the art of cryptographic algorithms as deployed for securing computing networks. While it has been argued that the design of efficient cryptographic algorithms is the "easy" part of securing a large scale network, it seems that very often security problems are identified in algorithms and their implementations. This article discusses the state of the art for a broad range of cryptographic algorithms that are used in networking applications.

Keywords: cryptographic algorithms, network security, block ciphers, stream ciphers, MAC algorithms, hash functions

1 Introduction

The first boom in cryptography can be attributed to the introduction of wireless data communications at the beginning of the 20th century [28]: it is clear that wireless communications are as easy to read for an adversary as for the legitimate receiver. There is also the mistaken perception that intercepting wired communications is really difficult; while the introduction of optical communications has raised the threshold, a well motivated opponent can also bypass this hurdle. From the 1960s, dedicated or switched wired networks were introduced for computer networks. Only military, governmental and financial communications were encrypted; until the early 1990s this encryption was mostly implemented in expensive hardware at the data link layer. The development of the world wide web resulted in broad use of cryptography for e-commerce and business applications. The underlying enabling technologies are inexpensive fast software cryptography and open security protocols such as TLS (SSL), SSH and IPsec as introduced in the second half of the 1990s. In spite of this development, only a small fraction of the Internet traffic is encrypted. Most of this encryption is situated at the network or transport layer; the communication is protected end-to-end (e.g., from the browser in the client to the web server), from gateway to gateway (for a VPN based on IPsec using tunnel mode) or from client to gateway (e.g., a VPN for remote access to company networks). In the last decade we have witnessed an explosion of wireless data networks, including Wireless LANs

(WLAN, IEEE 802.11), Personal Area Networks (PANs such as Bluetooth or IEEE 802.15, Zigbee or IEEE 802.15.4, and Ultrawideband or IEEE 802.15.4a) and Wireless Metropolitan Area Networks (WiMAX or IEEE 802.16). All these technologies have been introduced with cryptographic security at the link layer; the early solutions are typically not very robust. In addition mobile data communication is growing on the evolving GSM mobile phones using technologies such as GPRS and EDGE as on the third generation mobiles phones such as 3GSM.

End to end protection of voice communication is a relatively recent phenomenon. The main reason has been technological limitations, but there is also a significant legal barrier, since governments want to maintain the capability to perform wiretaps for law enforcement and national security purposes. Analog voice scramblers do not offer a very high security level. The US delegation in the 1945 Yalta conference brought along very voluminous devices for digital voice encryption; apparently they were never used, a.o. for the poor quality. Efficient digital coding of voice for mass market products arrive in the 1980s: secure digital phones (e.g. the STUs) became available, but outside the government and military environment they were never successful. However, today Voice over IP (VoIP) technologies result in widespread end-to-end security based on software encryption. The first analog mobile phones provided no or very weak security, which resulted in serious embarrassment (e.g., the private conversations of Prince Charles being exposed or the eavesdropping of the Soviet mobile communication systems by the US). The European GSM system designed in the late 1980s provided already much better security, even if many flaws remain; these flaws did not stop the system: in 2010 there are more than 4 billion GSM and WCDMA-HSPA subscribers. The GSM security flaws have been resolved in the 3GSM system, but even there no end-to-end protection is provided. The current generation of smart phones users can clearly run software (such as Skype) with this capability.

This short article tends to briefly describe the situation in terms of cryptographic algorithms used in communication networks. In Sect. 2 we present an update on hash functions, stream ciphers, block ciphers and their modes. Section 3 focuses on public key algorithms and Sect. 4 presents the conclusions.

2 Symmetric Primitives

In this section, we discuss the following symmetric primitives: block ciphers, stream ciphers, MAC algorithms, hash functions and modes for authenticated (or unforgeable) encryption.

2.1 Block Ciphers

Block ciphers are a flexible building block for many cryptographic applications. This includes the original goal of encryption (in CBC, CFB, OFB or CTR mode), but they can also be used to construct MAC algorithms (cf. Sect. 2.3), hash functions (cf. Sect. 2.4), pseudo-random functions and one-way functions.

The DES algorithm was published by the US government in the 1970s; it is a block cipher with a 64-bit block length and a 56-bit key. In spite of initial controversy around its design, the deciding factors in the success of the DES algorithm were the standardization by the US government and the generous licensing conditions. However, in the 1990s it became obvious that the 56-bit key size was no longer adequate.[1] The financial world started moving towards two key triple-DES in the late 1990s; this move was completed around 2006, a few years later than planned. In 2004 NIST (National Institute of Standards and Technology, US) announced that DES was no longer adequate and published a triple-DES specification [72]; two-key triple-DES is approved until 2009, while three-key triple-DES is deemed to be adequate until 2030. The modes for triple-DES have been defined in ANSI X9.52 [2]. The main reason for the limited lifetime of the two-key triple-DES variant is the attack by Wiener and van Oorschot [95] that requires 2^{80} time when 2^{40} known plaintexts are available; this is not a concern for the financial sector, as keys are typically changed frequently and messages are very short. On the other hand, three-key triple-DES is very vulnerable to a related-key attack [58]; in this attack an opponent obtains the encryption of a plaintext P under a key K and a key $K \oplus \Delta$ for a constant Δ. In most contexts such an attack is not feasible, but an exception is applications that use control vectors [68].

In 1997, NIST started an open competition to find a replacement for the DES. The AES algorithm has a block of length of 128 bits, and should support keys lengths of 128, 192 and 256 bits. In October 2000 NIST selected the Rijndael algorithm (designed by the Belgian cryptographers Vincent Rijmen and Joan Daemen) as the AES algorithm [24, 39]. In 2003, the US government announced that it would also allow the use of AES for secret data, and even for top secret data; the latter applications require key lengths of 192 or 256 bits. AES is a rather elegant and mathematical design, that among the five finalists offered the best combination of security, performance, efficiency, implementability and flexibility. AES allows for compact implementations on 8-bit smart cards (36 bytes of RAM), but also highly efficient implementations on 32-bit architectures (15 cycles/byte on a Pentium III and 7.6 cycles/byte on a Core 2 [55]). Moreover, hardware implementations of AES offer good trade-offs between size and speed. AES has been taken up quickly by many standards and implementations; in May 2010 more than 1300 AES implementations have been validated by the US government.

So far, AES has resisted all shortcut attacks, including algebraic attacks. In 2009, it was demonstrated by Biryukov and Khovratovich [11] that AES-192 and AES-256 are vulnerable to related-key attacks: the attack on AES-256 requires 4 related keys and 2^{119} encryptions, which is much less than 2^{256}. These attacks indicate that they key schedule of AES should have been stronger; on the other

[1] A US$ 1 million machine today would recover a DES key in a few seconds – the same design would have taken 3 hours in 1993 [103].

hand, they clearly do not form a practical threat and one can easily defend against them by not allowing any key manipulations or by hashing a key before use. It is also worth to point out that it is not possible to design a cipher that is secure against *any* related key attack.

In 2010 Dunkelman *et al.* [31] have published a related key attack on the 64-bit block cipher KASUMI (that is standardized for GSM under the name A5/3 and that is also used for encryption in 3GPP); the attack requires 4 related keys, 2^{26} plaintexts, 2^{30} bytes of memory and time 2^{32}; while these complexities are rather low, the attack cannot be applied to KASUMI as deployed in current mobile networks.

The most powerful attacks against AES and other block ciphers have not been pure mathematical attacks, but timing attacks based on cache effects – this kind of attack applies in principle to any cryptographic algorithm implementation that uses tables (see e.g. [9, 76, 94]). This attack is one of the reasons why Intel has to add dedicated AES instructions to its processors from 2010 onwards [44]; these instructions also boost the performance of AES to about 0.75 cycles/byte (in decryption mode). Note that the fast implementation of AES of Kaš̈per and Schwabe [55] is bitsliced and hence not vulnerable to cache-based attacks.

2.2 Stream Ciphers

Because of their low implementation cost, additive stream ciphers have been the work horse of symmetric cryptography until the 1980s. They take as input a short secret key and a public initialization value IV and stretch this to a long string that can be simply added to the plaintext to yield the ciphertext. This implies that the encryption transformation is very simple but depends on the location in the plaintext. Hardware oriented stream ciphers typically operate on short data units (bits or bytes) and have a small footprint. The initialization value IV serves for resynchronization purposes. Both the IV and the internal memory need to be sufficiently large to resist time-memory-data tradeoffs (see for example [46, 62]).

From the 1960s to the late 1980s, most stream ciphers were based on Linear Feedback Shift Registers (LFSRs) that are optimal for hardware implementations (see for example Rueppel [87] and Menezes *et al.* [71]). However, it has become clear that most LFSR-based stream ciphers are much less secure than expected; powerful new attacks include fast correlation attacks [70] and algebraic attacks [23]. Notable cryptanalytic successes are the attack by Barkan and Biham [3] on A5/1 (the stream cipher used in GSM) and the attack by Lu *et al.* [65] on E0 (the stream cipher used in Bluetooth). Both attacks are realistic attacks on widely used algorithms.

RC4 has been designed in 1987 by Rivest for efficient software encryption on 8-bit machines. RC4 was a trade secret, but leaked out in 1994; it is currently still implemented in browsers (SSL/TLS protocol). While several statistical weaknesses have been identified in RC4 [40, 77], the algorithm seems to resist key recovery attacks.

In the last decade, fast stream ciphers have been proposed that are oriented towards 32-bit and 64-bit processors. Two stream ciphers that have been included into the ISO standard are MUGI [100] and SNOW [33]; a strengthened variant of SNOW has been selected as backup algorithm for 3GSM. Between 2004 and 2008 the EU Network of Excellence ECRYPT [32] has organized an open competition eSTREAM with as goal to identify promising stream ciphers that are either very fast in software (128-bit key and 64 or 128-bit IV) or that offer a low footprint in hardware (80-bit key and 32 or 64-bit IV). During the four years of the competition, dozens of stream ciphers have been broken. The competition has resulted in a portfolio with four software-oriented ciphers with a performance of 3-10 cycles/byte (HC-128, Rabbit, Salsa20/12 and Sosemanuk); three hardware-oriented ciphers are recommended (Grain, Mickeyv2, and Trivium). An important conclusion from the eSTREAM project is that for very low footprint implementations, 64-bit block ciphers are more efficient; however, if one desires a very high performance implementation with a low hardware cost, the hardware-oriented stream ciphers offer an improvement with a factor of two to four over block ciphers. More details on the eSTREAM competition can be found in [84].

2.3 Message Authentication Codes (MACs)

Message Authentication Codes are used to authenticate messages between parties that share a secret key. MACs are widely use in networks, because they are more efficient in terms of performance and memory than digital signature schemes. The most widely used constructions are derived from block ciphers or hash functions.

The most popular MAC algorithm for financial transactions is still CBC-MAC. Initially, variants based on DES were used; these have been migrated to triple-DES variants. AES is gradually replacing DES for this application (cf. Sect. 2.1).

The CBC-MAC construction based on an n-bit block cipher can be described as follows. First the input string is padded to a multiple of the block length, and the resulting string is divided into t n-bit blocks x_1 through x_t.

$$c_1 := E_k(x_0) \tag{1}$$

$$c_i := E_k(x_i \oplus c_{i-1}), \quad 1 < i \leq t \ . \tag{2}$$

Here $\oplus$ denotes the bitwise exclusive-or operation. Note that – unlike in CBC encryption – no IV value should be used. The recommended variant for use with DES is the ANSI retail MAC [1]: it computes the MAC value with two independent keys k and k': $\mathrm{MAC}_k(x_0 \ldots x_t) = E_k\left(E_{k'}(c_t)\right)$. For AES, EMAC is the preferred construction: $\mathrm{MAC}_k(x_0 \ldots x_t) = E_{k'}(c_t)$. Here k' is a key derived from k. An even simpler scheme is LMAC; it uses the key k' for the last encryption ($i = t$).

NIST has published yet another variant under the name of CMAC [73] (CMAC was previously called OMAC [53], which is an optimization of XCBC [14]).

CMAC modifies the last computation in CBC-MAC by exoring $k2$ or $k3$ to x_t. The key $k2$ is chosen when the last block x_t requires no padding (i.e., it is of length n), while $k3$ is chosen otherwise. The keys $k2$ and $k3$ are computed as $k2 = \text{`2'} \cdot E_k(0^n)$ and $k3 = \text{`4'} \cdot E_k(0^n)$ where 0^n denotes the n-bit all zero string, '2' and '4' are two elements of the finite field F_{2^n}, and "·" represents multiplication in the finite field F_{2^n}.

On the Internet, HMAC is by far the most popular construction [5]; in the light of the attacks on MD4 and MD5 (cf. Sect. 2.4), the HMAC security analysis has been refined by Bellare [4]. The state of the art in cryptanalysis is that HMAC-MD4 has been broken by Leurent *et al.* [41]; their attack requires 2^{88} chosen texts and 2^{95} computations. Some doubts have been cast on HMAC-MD5 [21, 59]; the best known attack on HMAC-MD5 is a related key attack that requires 2^{51} chosen plaintexts and 2^{100} time (see also [99]). For the time the security margin offered by HMAC-SHA-1 is acceptable.

In the past five years there has been a growing interest in unconditionally secure MAC algorithms. They were introduced as authentication codes by Simmons [92] and more practical constructions were known as universal hash functions (following Carter and Wegman [101]). If they are combined with a block cipher (such as AES) or a pseudo-random function (such as HMAC), the unconditional security is lost, but they result in MAC algorithms that are very efficient and elegant. UMAC [13] is about 10 times faster than CBC-MAC based on AES or HMAC-SHA-1, but it offers a limited key agility and has a rather large Random Access Memory (RAM) requirement; moreover, Handschuh and Preneel have demonstrated [45] that for a large class of MAC algorithms based on universal hash functions (including UMAC) a few forgeries lead to efficient key recovery. Bernstein's Poly1305-AES [9] is one of the constructions based on polynomial universal hashing. It is only three times faster than AES, but it has a better key agility than UMAC and requires less RAM; it seems also less vulnerable to key recovery attacks.

2.4 Hash Functions

Cryptographic hash functions are a widely deployed primitive for message authentication. They compress strings of arbitrary lengths to strings of fixed lengths (typically between 128 and 256 bits). Cryptographic hash functions need to satisfy the following three security properties [71, 79]:

- preimage resistance: it should be hard to find a preimage for a given hash result;
- 2nd preimage resistance: it should be hard to find a 2nd preimage for a given input;
- collision resistance: it should be hard to find two different inputs with the same hash result.

For an ideal hash function with an n-bit result, finding a (2nd) preimage requires approximately 2^n hash function evaluations. On the other hand, finding a collision requires only $2^{n/2}$ hash function evaluations (as a consequence of the birthday paradox). Collision resistance implies 2nd preimage resistance, but the formal relation between these definitions is more complex and subtle than one would expect (see Rogaway and Shrimpton [86]). In practice on requires also other properties such as indifferentiability from a random oracle [22], and pseudo-randomness (this assumes that a secret key is part of the input).

The main application of hash function is digital signature schemes, in which one signs the hash value of a message rather than the message itself. Digital signatures are used in some key establishment protocols to bind a protocol message to an entity. Hash functions can also be used to construct MAC algorithms; the most popular construction of this type is HMAC (cf. Sect. 2.3). HMAC constructions are also used for deriving symmetric keys in protocols such as Diffie-Hellman. In practice HMAC is used with hash functions such as MD5, SHA-1 and RIPEMD-160. In the SSL/TLS protocol, a hash function is used at the end of the handshake protocol (in which the cipher suites are negotiated) to confirm the integrity (TLS version 1.0/1.1 uses the concatenation of MD5 and SHA-1, while in TLS version 1.2 a single hash function is used).

In the last decade, a number of structural weaknesses have been identified in hash functions; these weaknesses are related to the way cryptographic hash functions are constructed from smaller building blocks. Most constructions use a simple iteration, and are therefore called iterated hash functions. The most remarkable attack is a result by Joux [54] who shows that if finding a collision for an iterated hash function takes time T (for an ideally secure hash function $T = 2^{n/2}$), one can find 2^s strings hashing to a single value in time $s \cdot T$. As an example, finding a billion messages that all hash to the same result requires only thirty times the effort to find a single collision. This result has the surprising corollary that the concatenation of two iterated hash functions ($g(x) = h_1(x)||h_2(x)$) is only as strong as the strongest of the two hash functions (even if both are independent). If h_i is a hash function with an n_i-bit result ($i = 1, 2$ and w.l.o.g. $n_1 \geq n_2$), finding a collision for g requires time at most $n_1 \cdot 2^{n_2/2} + 2^{n_1/2} \ll 2^{(n_1+n_2)/2}$ and finding a preimage or 2nd preimage for g requires time at most $n_1 \cdot 2^{n_2/2} + 2^{n_1} + 2^{n_2} \ll 2^{n_1+n_2}$. If either of the functions is weak, the attacks may work better. This attack is particularly relevant since weaknesses have been discovered in several widely used hash functions (cf. infra) and the concatenation construction has been proposed as a robust solution (e.g. in SSL/TLS). It seems that once the collision resistance of our current iterated hash functions breaks down, the other security properties are also undermined.

Until recently, the most widely used hash functions were MD5 and SHA-1. MD5 is a 128-bit hash function designed by Rivest in 1991 [82]; it is a strengthened version of MD4. MD5 was one of the first cryptographic algorithms that was designed to be fast on 32-bit processors in software. Early cryptanalytic results by den Boer and Bosselaers [26] and Dobbertin [30] indicated that finding collisions for MD5 would require less than 2^{64} operations; in spite of the fact that cryptographers advised against using MD5, the algorithm has been widely deployed. The first collisions for MD5 were announced in 2004 by Wang *et al.* [98], who were able to push the limits on differential attacks by introducing some innovative cryptanalytic techniques; their attack required time 2^{39}, which corresponds to a few hours on a PC. Since then the attack has been further optimized; the best collision search algorithm known today requires milliseconds [93]. While this represents a major breakthrough, it is important to note that with about US\$100 000 of hardware, a brute-force collision search for MD5 (or any 128-bit hash function of comparable cost) should take a few days with the design of van Oorschot and Wiener [96].

In 1995, NIST has published SHA-1 [37]; it is a strengthened version of SHA, which was standardized two years earlier [36] (SHA is now called SHA-0 by some researchers). Both SHA(-0) and SHA-1 have a 160-bit result. While SHA-1 is slower but more secure than MD5, it became very popular for applications that require long term security. In 2005, Wang *et al.* [97] have published a collision search algorithm for SHA-1 that requires only 2^{69} steps, which is 2000 times faster than a brute force collision search. Five years later, several researchers have announced improvements (sometimes even very spectacular ones), but so far none of these attacks has materialized. In 2005 Joux *et al.* [54] found collisions for SHA(-0) with complexity 2^{51}. Today the best collision attack for SHA-0 by Manuel and Peyrin [67] takes only 2^{33} steps. The implications of the attack on SHA(-0) are limited, since this algorithm is not deployed.

The collision attacks on MD4 and MD5 are quite unusual in the sense that they are extremely efficient. However, so far their practical implications have been limited, as very few applications use digital signatures and very few applications require collision resistance. In December 2008, Sotirov et al. [93] created a rogue CA certificate using MD5, which allows them to impersonate any website on the Internet. This attack required cryptanalytic improvements beyond simple collision search. Only after this attack, several Certification Authorities decided to remove MD5 from their offerings. While there is substantial progress with preimage attacks on MD4 and MD5, these attacks are far from practical. Leurent [64] has shown that preimages for MD4 can be found in 2^{102} steps, and the preimage attack by Sasaki and Aoki [89] on MD5 has complexity 2^{123}.

RIPEMD-160 [19] could act as a replacement for SHA-1; it seems to resist all cryptanalytic efforts. NIST has also a series of standards that offer longer hash results: SHA-256, SHA-224, SHA-384 and SHA-512 [38], which are known under the common name SHA-2. Cryptanalysis of the SHA-2 family suggests that this second generation functions has a substantial security margin against collision attacks (the results by Indesteege et al. [48] and Sanadhya and Sarkar [88] can only break 24 out of 64 steps of SHA-256). A third alternative is Whirlpool, a design by Rijmen and Barreto [51] based on the design principles of AES. For the most recent status of attacks on Whirlpool, see [61]. All these hash functions have been standardized by ISO in IS 10118–3 [51], together with SHA-1.

NIST is currently running an open competition for a new hash function standard that will be called SHA-3. Sixty-four submissions have been received, 14 of which are currently being evaluated in the second round. It is expected that NIST will announce the winner by mid 2012. For more details on the SHA-3 competition and on the state of hash functions, see [79].

2.5 Authenticated or Unforgeable Encryption

Most applications need a secure channel between sender and receiver; such a channel requires both confidentiality and data authentication. In the 1980s and 1990s, separate primitives were introduced for each of these properties. However, it is not so hard to show that confidentiality protection without data authentication can lead to serious problems; in particular, such a scheme is vulnerable to a chosen ciphertext attack in which the opponent uses decryption queries to learn information on the plaintext. Practical chosen ciphertext attacks have been demonstrated by several authors; we must mention the attack by Canvel *et al.* on SSL/TLS [20] and the attack by Degabriele and Paterson on IPsec [25].

The first approach to achieve both properties was to introduce redundancy to the plaintext before encryption in order to achieve both goals, but this is clearly not adequate. A first formalization of unforgeable encryption was published by Katz and Yung [56]. Bellare and Namprempre [6] showed that if the MAC algorithm satisfies a strong security requirement (namely strong unforgeability), the best generic solution is to apply a MAC algorithm to the ciphertext (the so-called Encrypt-then-MAC model), which is the option chosen by IPsec. Other alternatives (MAC-then-Encrypt of SSL/TLS and Encrypt and MAC of SSH) can also be shown to be secure, but they require a specific rather than a generic analysis (e.g., taking into account the specific encryption mode).

The above schemes require both an encryption algorithm and a MAC algorithm. Jutla showed that it was possible to achieve both properties at a much lower cost; for this purpose he introduced in 2000 two modes, the IACBC (Integrity-Aware Cipher Block Chaining) and IAPM (Integrity-Aware Parallelizable Mode). Gligor and Donescu proposed the XCBC and XECB schemes in [43]. Rogaway *et al.* [85] introduced an optimized version of IAPM called the OCB mode (Offset CodeBook). These schemes require an overhead of less than 10% over CBC encryption and offer some attractive features; for example, some of them are fully parallellizeable. An important non-technical disadvantage is that all these schemes are encumbered by patents, which has been a barrier to their adoption.

As a consequence of this patent issue, several alternative schemes have been introduced that are slower than these schemes, but that are free. NIST and ISO have standardized a combination of the counter mode with a polynomial based authentication (the Galois Counter Mode or GCM [69, 75]) and with CBC-MAC (the Counter with CBC-MAC mode [102, 74]). For a more detailed overview of authenticated encryption schemes, see the overview article by Black [12] and the ECRYPT II report [32].

3 Public Key Algorithms

In network security, public key algorithms are only used for the establishment of session keys and for the mutual authentication of the parties. The main reason is that public key operations are two or three orders of magnitude slower than symmetric key primitives. Moreover, the block lengths and overhead are substantially larger. Public key algorithms need to be integrated into a protocol such as the Station-to-Station protocol [29]; more elaborate variants of this protocol have been standardized for SSL/TLS (RFC 5246) and for IPsec (IKEv2 in RFC 4306). The details of these protocols fall outside the scope of this article.

3.1 RSA

RSA, invented by Rivest, Shamir and Adleman in 1978 [83] is by far the most widely used public key algorithm (the RSA patent has expired in 2000). The RSA encryption operation is written as $C = P^e \bmod N$ and the decryption is computed as $P = C^d \bmod N$. Here the encryption and decryption exponent are related by $e \cdot d = 1 \bmod \operatorname{lcm}(p-1, q-1)$, with $N = p \cdot q$. The security of RSA is based on the fact that it is relatively easy to find two large prime numbers p and q, but no efficient methods are known to factor their product N. Note that the security of RSA is based on the fact that extracting random modular eth roots mod$N$ is hard. This problem could be easier than factoring N (it cannot be harder); surprisingly, whether or not it is easier is still an open problem.

The best known algorithm to factor an RSA modulus N is the General Number Field Sieve (GNFS). Lenstra and Verheul have related the complexity of GNFS to breaking symmetric keys and computing discrete logarithms in [63] (see also the ECRYPT II report on this topic [32]). The current factoring record (achieved in January 2010) is 768 bits [60]. The recommended minimum size for an RSA modulus today is 1024 bits; factoring such a modulus requires approximately 2^{72} steps. Shamir and Tromer [90] proposed in 2003 a hardware design that would need an R&D effort of US$20 M. The hardware cost to factor a 512-bit modulus in ten minutes would be US$ 10 000; a 768-bit modulus could be factored with a similar budget in 95 days; factoring a 1024-bit modulus in 1 year would require a hardware investment of US$ 10 M. Note that these cost estimates do not include the linear algebra step. These estimates show that for long-term security (10-15 years), an RSA modulus of 2048 bits or more is recommended.

Textbook RSA has other weaknesses (see [18] for details). For example, RSA for small arguments is not secure: -1, 0 and 1 are always fixed points and if $P^e < N$ extracting a modular eth root simplifies to extracting a natural eth root, which is an easy problem. In addition, RSA is multiplicative, which means that the product mod$N$ of two ciphertexts will decrypt to the product of the corresponding plaintexts.

The standard PKCS#1v1.5 specifies a padding method for encryption and signing with the RSA algorithm. For encryption, the format consists of the following sequence: a byte equal to `00`, a byte equal to `02`, at least 8 non-zero padding bytes, a byte `00`, and the plaintext. Note that the RSA assumption states that extracting *random* modular eth roots is hard, which means that one should map the plaintext space in a uniform way to the interval $[0, n[$; it is clear that PKCS#1v1.5 is quite far from this goal. This has been exploited by Bleichenbacher [15] to recover the plaintext corresponding to a selected ciphertext using a chosen ciphertext attack (in which encryptions of different but related ciphertexts are obtained); more specifically, Bleichenbacher's attack only needs to know whether the plaintext is of the right format (it is based on the error messages). In 1993, Bellare and Rogaway published the OAEP (Optimal Asymmetric Encryption) transform, together with a security proof [7]. This proof essentially states that if someone can decrypt a challenge ciphertext without knowing the secret key, he can extract random modular eth roots. The proof is in the random oracle model, which means that the hash functions used in the OAEP construction are assumed to be perfectly random. However, seven years later Shoup pointed out that the proof was wrong [91]; the error has been corrected for by Fujisaki *et al.* in [42], but the resulting reduction is not very meaningful, that is, the coupling between the two problems is not very tight in this new proof. Moreover, Manger showed that a careful implementation is necessary, since otherwise a chosen ciphertext attack based on error messages may still apply [66]. Currently the cryptographic community believes that the best way of using RSA is the RSA-KEM mode [80]: this is a so-called *hybrid* mode in which RSA is only used to transfer a session key, while the plaintext is encrypted using a symmetric algorithm with this key.

For RSA PKCS#1v1.5 signatures, no practical attack is known, even if this padding format is again very far from random. The RSA signing operation is applied to the following sequence: a byte equal to `00`, a byte equal to `01`, a series of bytes equal to `FF`, a byte `00`, and the hash value (with some ASN.1 prepended). At the rump session of Crypto 2006, Bleichenbacher showed that many implementations of RSA signature verifications stop at the end of the hash value. This opens the possibility to append a large random string S (and shorten the series of `FF` bytes accordingly). It is very easy to choose S such that the complete string is a perfect cube, and extracting cube roots over the integers is easy. This means that one can forge any signature for $e = 3$ without knowing the private key; even better, this forged signature works for any modulus N that is large enough. A variant of the attack is based on the fact that some verification software ignores the content of the ASN.1 string. These attacks can be precluded by implementing a correct verification, which consists of checking that the hash value is right aligned or alternatively by re-generating the whole block as the signer does and checking that it is correct. The problem is however that as a signer may not be able to influence the verification software, hence it is better to increase the verification exponent to $2^{16} + 1$. Implementations that were reported to be vulnerable to this problem include OpenSSL, Mozilla NSS, and GnuTLS. A better solution is to use RSA-PSS [8], which has been included together with OAEP in PKCS#1 v2.1. Even if the scheme dates back to 1996 and the standard to 2002, so far implementors seem to be reluctant to upgrade to the more robust algorithms.

For performance reasons, the RSA private key operations (decryption and signing) are often executed using the Chinese remainder theorem. This means that they are computed $\bmod p$ and $\bmod q$ and that both results are combined to recover the result $\bmod N$. One of the most important vulnerabilities of RSA in practice is the observation by Boneh *et al.* [17]: if a transient fault is introduced in the calculation $\bmod p$ or $\bmod p$ (but not both), one can recover p and q. Making an implementation robust against these powerful fault attacks is non-trivial.

An important lesson that can be drawn from this is that it is surprisingly difficult to use RSA correctly: it has taken the cryptographic community more than 20 years to learn how to do this. The most efficient solutions still rely on the random oracle model, and it is an important problem how one can use RSA efficiently without this assumption.

3.2 Elliptic Curve Cryptography (ECC)

Elliptic curve cryptography (ECC) is a public-key primitive that is increasingly important as alternative to RSA. The standards (e.g., [52, 47]) support both elliptic curves over F_p with p prime and F_{2^m} with m prime. The first curves can take advantage from an arithmetic coprocessor for RSA if available, while the latter allow for very compact hardware implementations.

An important advantage of elliptic curves are the shorter key lengths. Based on the best known algorithms today, one can estimate that 160-bit elliptic curves correspond to 1248-bit RSA, and 224-bit elliptic curves correspond to 2432-bit RSA (see the ECRYPT II report [32]). For these bit-lengths, signing is about five (resp. 20) times faster with elliptic curves, but verifying a signature is seven (resp. five) times faster with RSA. Moreover, very compact hardware implementations of ECC have been developed.

ECC was proposed in 1985; for the first 15 years the market was reluctant to adopt this new and more complex primitive. However, in the past five years ECC has been selected by the governments of Austria, Germany, Switzerland and the USA and are gaining more widespread acceptance. The main attraction lies clearly in the shorter key lengths; this advantage over RSA will grow larger over time.

4 Conclusions

During the past decade, the AES has become the de facto standard for encrypting network data. HMAC-MD5 and HMAC-SHA-1 are the most common algorithms used for message authentication. We see a gradual evolution towards using mechanisms for authenticated or unforgeable encryption, which combine encryption and data authentication in one operation. Those modes require a redesign of the protocol. In this context, HMAC is increasingly replaced by CBC-MAC based on AES or a polynomial hash function; the latter is substantially faster but perhaps a bit less robust. Wireless networks still use older block ciphers or stream ciphers; 3G networks offer data authentication based on MAC algorithms.

For public key algorithms the evolution has been much slower. RSA and Diffie-Hellman based protocols over F_p are getting more and more competition from ECC, in particular for low footprint or low power environments. The relatively smaller keys for ECC is a key factor in this development.

Side channel attacks have become an important area of research: they currently strongly influence hardware and software implementations, but at the cost of a decreased performance. One can expect that in the future some algorithms will be re-designed from scratch so that implementing these algorithms in a secure way is easier.

In addition to new attacks, new security proofs and models have been developed, that increase our understanding in areas such as modes for confidentiality and authenticated encryption and padding methods for RSA and ECC.

In both cases (new attacks and new models and designs), there is a need for efficient and secure procedures to upgrade and retire cryptographic algorithms. However, even if we live in a world in which the environment can change in days or months, replacing a cryptographic algorithm still takes many years. System designers need to build systems that are agnostic to the cryptographic algorithm and that allow for fast and secure key length and algorithm upgrades.

Acknowledgements. This work was partially funded by the European Commission through the IST Programme under Contract ICT-2007-216676 ECRYPT II and by the Belgian Government through the IUAP Programme under contract P6/26 BCRYPT.

References

1. ANSI X9.19, *Financial Institution Retail Message Authentication,* American Bankers Association, August 13, 1986.
2. ANSI X9.52, *Triple Data Encryption Algorithm Modes of Operation,* American Bankers Association, 1998.
3. E. Barkan, E. Biham, N. Keller, Instant ciphertext-only cryptanalysis of GSM encrypted communication, *Advances in Cryptology, Proceedings Crypto'03, LNCS 2729*, D. Boneh, Ed., Springer, Heidelberg, 2003, pp. 600–616.
4. M. Bellare, New proofs for NMAC and HMAC: Security without collision-resistance, *Advances in Cryptology, Proceedings Crypto'06, LNCS 4117*, C. Dwork, Ed., Springer, Heidelberg, 2006, pp. 602–619.
5. M. Bellare, R. Canetti, H. Krawczyk, Keying hash functions for message authentication, *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109*, N. Koblitz, Ed., Springer, Heidelberg, 1996, pp. 1–15.
6. M. Bellare, C. Namprempre, Authenticated encryption: Relations among notions and analysis of the generic composition paradigm, *Advances in Cryptology, Proceedings Asiacrypt'00, LNCS 1976*, T. Okamoto, Ed. (Springer, Heidelberg, 2000) 531–545.

7. M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols, *Proceedings ACM Conference on Computer and Communications Security* (ACM Press 1993) 62–73.
8. M. Bellare, P. Rogaway, The exact security of digital signatures – How to sign with RSA and Rabin, *Advances in Cryptology, Proceedings Eurocrypt'96, LNCS 1070*, U. Maurer, Ed., Springer, Heidelberg, 1996, pp. 399–416.
9. D.J. Bernstein, The Poly1305-AES message-authentication code, *Fast Software Encryption, LNCS 3557*, H. Gilbert and H. Handschuh, Eds. (Springer, Heidelberg, 2005) 32–49.
10. D.J. Bernstein, Cache-timing attacks on AES, preprint, 2005, `http://cr.yp.to/papers.html#cachetiming`
11. A. Biryukov, D. Khovratovich, "Related-key cryptanalysis of the full AES-192 and AES-256," *Advances in Cryptology, Proceedings Asiacrypt'09, LNCS 5912*, M. Matsui, Ed., Springer, Heidelberg, 2009, pp. 1–18.
12. J. Black, Authenticated encryption, *Encyclopedia of Cryptography and Security* H. van Tilborg, Ed., Springer, Heidelberg, 2005, pp. 11–21.
13. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, P. Rogaway, UMAC: Fast and secure message authentication, *Advances in Cryptology, Proceedings Crypto'99, LNCS 1666*, M.J. Wiener, Ed., Springer, Heidelberg, 1999, pp. 216–233.
14. J. Black, P. Rogaway, CBC-MACs for arbitrary length messages, *Advances in Cryptology, Proceedings Crypto'00, LNCS 1880*, M. Bellare, Ed. (Springer, Heidelberg, 2000) 197–215.
15. D. Bleichenbacher, Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1, *Advances in Cryptology, Proceedings Crypto'98, LNCS 1462*, H. Krawczyk, Ed., Springer, Heidelberg, 1998, pp. 1–12.
16. D. Bleichenbacher, Forging some RSA signatures with pencil and paper, *Presented at the Rump Session of Crypto 2006.*
17. D. Boneh, R. DeMillo, R. Lipton, On the importance of checking cryptographic protocols for faults, *Advances in Cryptology, Proceedings Eurocrypt'97, LNCS 1233*, W. Fumy, Ed., Springer, Heidelberg, 1997, pp. 37–51.
18. D. Boneh, A. Joux, P.Q. Nguyen, Why textbook ElGamal and RSA encryption are insecure, *Advances in Cryptology, Proceedings Asiacrypt'00, LNCS 1976*, T. Okamoto, Ed. (Springer, Heidelberg, 2000) 30–43.
19. A. Bosselaers, H. Dobbertin, B. Preneel, The RIPEMD-160 cryptographic hash function, *Dr. Dobb's Journal*, Vol. 22, No. 1, January 1997, pp. 24–28.
20. B. Canvel, A.P. Hiltgen, S. Vaudenay, M. Vuagnoux, "Password interception in a SSL/TLS Channel," *Advances in Cryptology, Proceedings Crypto'03, LNCS 2729*, D. Boneh, Ed., Springer, Heidelberg, 2003, pp. 583–599.
21. S. Contini, Y.L. Lin, Forgery and partial key recovery attacks on HMAC and NMAC using hash collisions *Advances in Cryptology, Proceedings Asiacrypt'06, LNCS 4284*, X. Lai and K. Chen, Eds., Springer, Heidelberg, 2006, pp. 37–53.
22. J.-S. Coron, Y Dodis, C. Malinaud, and P. Puniya, "Merkle-Damgård revisited: how to construct a hash function," *Advances in Cryptology, Proceedings Crypto'05, LNCS 3621*, V. Shoup, Ed., Springer, Heidelberg, 2005, pp. 430–448.
23. N. Courtois, W. Meier, Algebraic attacks on stream ciphers with linear feedback, *Advances in Cryptology, Proceedings Eurocrypt'03, LNCS 2656*, E. Biham, Ed. (Springer, Heidelberg, 2003) 345–359.
24. J. Daemen, V. Rijmen, *The Design of Rijndael. AES – The Advanced Encryption Standard,* Springer, Heidelberg (2001).

25. J.P. Degabriele, K.G. Paterson, "Attacking the IPsec standards in encryption-only configurations," in *IEEE Symposium on Security and Privacy*, IEEE, 2007, pp. 335–349.
26. B. den Boer, A. Bosselaers, Collisions for the compression function of MD5, *Advances in Cryptology, Proceedings Eurocrypt'93, LNCS 765*, T. Helleseth, Ed., Springer, Heidelberg, 1994, pp. 293–304.
27. T. Dierks, E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.
28. W. Diffie, S. Landau, "Privacy on the Line. The Policy of Wiretapping and Encryption (2nd edition)," MIT Press, 2007.
29. W. Diffie, P.C. van Oorschot, M.J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes, and Cryptography*, **2**(2) 107–125 (1992)
30. H. Dobbertin, The status of MD5 after a recent attack, *CryptoBytes*, Vol. 2, No. 2, Summer 1996, pp. 1–6.
31. O. Dunkelman, N. Keller, A. Shamir, "A practical-time attack on the KASUMI cryptosystem used in GSM and 3G telephony," *Advances in Cryptology, Proceedings Crypto'10, LNCS*, T. Rabin, Ed., Springer, Heidelberg, 2010, in print.
32. EU Network of Excellence ECRYPT II, Yearly Report on Algorithms and Keysizes, 2009–2010, `http://www.ecrypt.eu.org`
33. P. Ekdahl, T. Johansson, A new version of the stream cipher SNOW, *Selected Areas in Cryptography, SAC'02, LNCS 2595*, K. Nyberg and H.M. Heys, Eds. (Springer, Heidelberg, 2003) 47–61.
34. Electronic Frontier Foundation, *Cracking DES, Secrets of Encryption Research, Wiretap Politics & Chip Design,* (O'Reilly & Associates, Sebastopol, 1998). Source code of the implementation described in the book can be downloaded from `https://www.cosic.esat.kuleuven.ac.be/des/`.
35. EU Directive 1999/93/EC, Community framework for electronic signatures, 13 December 1999.
36. FIPS 180, *Secure Hash Standard,* Federal Information Processing Standard (FIPS), Publication 180, NIST, U.S. Dept. of Commerce, May 11, 1993.
37. FIPS 180-1, *Secure Hash Standard,* Federal Information Processing Standard (FIPS), Publication 180-1, NIST, U.S. Dept. of Commerce, April 17, 1995.
38. FIPS 180-2, *Secure Hash Standard,* Federal Information Processing Standard (FIPS), Publication 180-2, NIST U.S. Dept. of Commerce, August 26, 2002 (Change notice 1 published on December 1, 2003).
39. FIPS 197, *Advanced Encryption Standard,* Federal Information Processing Standard, NIST, U.S. Dept. of Commerce, November 26, 2001.
40. S. Fluhrer, I. Mantin, A. Shamir, Weaknesses in the key scheduling algorithm of RC4, *Selected Areas in Cryptography, SAC'01, LNCS 2259,* S. Vaudenay and A. Youssef, Eds. (Springer, Heidelberg 2001) 1–24.
41. P.-A. Fouque, G. Leurent, P.Q. Nguyen, "Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5," *Advances in Cryptology, Proceedings Crypto'07, LNCS 4622*, A. Menezes, Ed., Springer, Heidelberg, 2007, pp. 13–30.
42. E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, RSA-OAEP is secure under the RSA assumption, *Advances in Cryptology, Proceedings Crypto'01, LNCS 2139*, J. Kilian, Ed., Springer, Heidelberg, 2001, pp. 260–274.
43. V.D. Gligor, P. Donescu, Fast encryption and authentication: XCBC encryption and XECB authentication modes, *Fast Software Encryption'01, LNCS 2355*, M. Matsui, Ed., Springer, Heidelberg, 2002, pp. 92–108.

44. S. Gueron, "Intel's new AES instructions for enhanced performance and security," *Fast Software Encryption'09, LNCS 5665*, O. Dunkelman, Ed., Springer, Heidelberg, 2009, pp. 51–66.
45. H. Handschuh, B. Preneel, "Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms," *Advances in Cryptology, Proceedings Crypto'08, LNCS 5157*, D. Wagner, Ed., Springer, Heidelberg, 2008, pp. 144–161.
46. J. Hong, P. Sarkar, New applications of time memory data tradeoffs, *Advances in Cryptology, Proceedings Asiacrypt'05, LNCS 3788*, B.K. Roy, Ed. (Springer, Heidelberg, 2005) 353–372.
47. IEEE P1363, *Standard Specifications for Public Key Cryptography*, 2000.
48. S. Indesteege, F. Mendel, B. Preneel, C. Rechberger, "Collisions and other nonrandom properties for step-reduced SHA-256," *Selected Areas in Cryptology – SAC 2008, LNCS 5381*, R. Avanzi, L. Keliher, and F. Sica, Eds., Springer, Heidelberg, 2009, pp. 276–293.
49. ISO/IEC 7816, *Information technology – Identification cards – Integrated circuit(s) cards with contacts – Part 4: Interindustry commands for interchange*, 1997.
50. ISO/IEC 9797, *Information technology – Security techniques – Message Authentication Codes (MACs), Part 1: Mechanisms using a block cipher*, 1999, *Part 2: Mechanisms using a hash-function*, 2000.
51. ISO/IEC 10118, *Information technology – Security techniques – Hash-functions, Part 1: General*, 2000, *Part 2: Hash-functions using an n-bit block cipher algorithm,* 2000, *Part 3: Dedicated hash-functions,* 2003. *Part 4: Hash-functions using modular arithmetic,* 1998.
52. ISO/IEC 14888-3, *Information technology – Security techniques – Digital signatures with appendix, Part 3: Certificate-based mechanisms*, 1998.
53. T. Iwata, K. Kurosawa, OMAC: One key CBC MAC, *Fast Software Encryption, LNCS 2887*, T. Johansson, Ed. (Springer, Heidelberg, 2003) 129–153.
54. A. Joux, "Multicollisions in iterated hash functions. Application to cascaded constructions," *Advances in Cryptology, Proceedings Crypto'04, LNCS 3512*, M.K. Franklin, Ed., Springer, Heidelberg, 2004, pp. 306–316.
55. E. Käsper, P. Schwabe, "Faster and Timing-Attack Resistant AES-GCM," *C Cryptographic Hardware and Embedded Systems, CHES 2009, LNCS 5747*, C. Clavier and K. Gaj, Eds., Springer, Heidelberg, 2009, pp. 1–17.
56. J. Katz, M. Yung, Unforgeable encryption and chosen ciphertext secure modes of operation, *Fast Software Encryption, LNCS 1978*, B. Schneier, Ed. (Springer, Heidelberg, 2001) 284–299.
57. C. Kaufman, "Internet Key Exchange (IKEv2) Protocol," RFC 4306, December 2005.
58. J. Kelsey, B. Schneier, D. Wagner, "Key-Schedule Cryptoanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES," *Advances in Cryptology, Proceedings Crypto'96, LNCS 1109*, N. Koblitz, Ed., Springer, Heidelberg, 1996, pp. 237–251.
59. J. Kim, A. Biryukov, B. Preneel, S. Hong, On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1 *Security in Communication Networks, LNCS 4116,* R. De Prisco, M. Yung, Eds., Springer, Heidelberg, 2006, pp. 242–256.
60. T. Kleinjung, K. Aoki, J. Franke, A.K. Lenstra, E. Thomé, J.W. Bos, P. Gaudry, A. Kruppa, P.L. Montgomery, D.A. Osvik, H. te Riele, A. Timofeev, P. Zimmerman, Factorization of a 768-bit RSA modulus, *Advances in Cryptology, Proceedings Crypto'10, LNCS*, T. Rabin, Ed., Springer, Heidelberg, 2010, in print.

61. M. Lamberger, F. Mendel, C. Rechberger, V. Rijmen, M. Schläffer, "Rebound distinguishers: results on the full Whirlpool compression function," *Advances in Cryptology, Proceedings Asiacrypt'09, LNCS 5912*, M. Matsui, Ed., Springer, Heidelberg, 2009, pp. 126–143.
62. J. Lano, Cryptanalysis and Design of Synchronous Stream Ciphers, *PhD Thesis, COSIC, K.U.Leuven*, June 2006.
63. A.K. Lenstra, E.R. Verheul, Selecting cryptographic key sizes, *J. Cryptology*, **14**(4) 255–293 (2001)
64. G. Leurent, "MD4 is not one-way," *Fast Software Encryption'08, LNCS 5086*, K. Nyberg, Ed., Springer, Heidelberg, 2008, pp. 412–428.
65. Yi Lu, W. Meier, S. Vaudenay, The conditional correlation attack: A practical attack on Bluetooth encryption, *Advances in Cryptology, Proceedings Crypto'05, LNCS 3621*, V. Shoup, Ed., Springer, Heidelberg, 2005, pp. 97–117.
66. J. Manger, A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS#1 v2.0, *Advances in Cryptology, Proceedings Crypto'01, LNCS 2139*, J. Kilian, Ed., Springer, Heidelberg, 2001, pp. 230–238.
67. S. Manuel, T. Peyrin, "Collisions on SHA-0 in one hour," *Fast Software Encryption'08, LNCS 5086*, K. Nyberg, Ed., Springer, Heidelberg, 2008, pp. 16–35.
68. S.M. Matyas, "Key Processing with Control Vectors," *J. Cryptology*, **3**(2) 113–136 (1991)
69. D. McGrew, J. Viega, The security and performance of the Galois/Counter Mode (GCM) of operation, *Progress in Cryptology – Indocrypt 2004, LNCS 3348*, A Canteaut and K. Viswanathan, Eds., Springer, Heidelberg, 2004, pp. 343–355. Full paper `http://eprint.iacr.org/2004/193/`
70. W. Meier, O. Staffelbach, Fast correlation attacks on stream ciphers, *J. Cryptology*, **1**(3) 159–176 (1989)
71. A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography* (CRC Press, 1997).
72. NIST Special Publication 800-67, *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher,* May 2004
73. NIST Special Publication 800-38B, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication,* May 2005.
74. NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality,* May 2004.
75. NIST Special Publication 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC,* November 2007.
76. D. Osvik, A. Shamir, E. Tromer, Cache attacks and countermeasures: The case of AES, *Topics in Cryptology – The Cryptographers' Track at the RSA Conference 2006, LNCS 3860,* D. Pointcheval, Ed. (Springer, Heidelberg 2006) 1–20. Extended version at `www.wisdom.weizmann.ac.il/~tromer/papers/cache.pdf`
77. S. Paul, B. Preneel, Analysis of non-fortuitous predictive states of the RC4 key stream Generator, *Progress in Cryptology, Indocrypt'04, LNCS 2904,* T. Johansson, S. Maitra, Eds., Springer, Heidelberg, 2003, pp. 30–47.
78. E. Petrank, C. Rackoff, CBC MAC for real-time data sources, *J. Cryptology*, **13**(3) 315–338 (2000)
79. B. Preneel, "The First 30 Years of Cryptographic Hash Functions and the NIST SHA-3 Competition," *Topics in Cryptology – CT-RSA 2010, LNCS 5985*, J. Pieprzyk, Ed., Springer, Heidelberg, 2010, pp. 1–14.

80. B. Preneel, A. Biryukov, C. De Cannière, S.B. Örs, E. Oswald, B. Van Rompay, L. Granboulan, E. Dottax, G. Martinet, S. Murphy, A. Dent, R. Shipsey, C. Swart, J. White, M. Dichtl, S. Pyka, M. Schafheutle, P. Serf, E. Biham, E. Barkan, Y. Braziler, O. Dunkelman, V. Furman, D. Kenigsberg, J. Stolin, J-J. Quisquater, M. Ciet, F. Sica, H. Raddum, L. Knudsen, M. Parker, *Final report of NESSIE, New European Schemes for Signatures, Integrity, and Encryption, LNCS* Springer, Heidelberg, in print.
81. B. Preneel, P.C. van Oorschot, MDx-MAC and building fast MACs from hash functions, *Advances in Cryptology, Proceedings Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer, Heidelberg, 1995, pp. 1–14.
82. R.L. Rivest, The MD5 message-digest algorithm, RFC 1321, April 1992.
83. R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications ACM*, Vol. 21, No. 2, 1978, pp. 120–126.
84. *New Stream Cipher Designs – The eSTREAM Finalists, LNCS 4986* M.J.B. Robshaw, O. Billet, Eds., Springer, Heidelberg, 2008.
85. P. Rogaway, M. Bellare, J. Black, T. Krovetz, OCB: A block-cipher mode of operation for efficient authenticated encryption, *ACM Conference on Computer and Communications Security*, ACM Press 2001, pp. 195–205.
86. P. Rogaway, T. Shrimpton, Cryptographic hash function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance, *Fast Software Encryption'04, LNCS 3017*, B.K. Roy and W. Meier, Eds., Springer, Heidelberg, 2004, pp. 371–388.
87. R.A. Rueppel, *Analysis and Design of Stream Ciphers*, Springer, Heidelberg, 1986.
88. S.K. Sanadhya, P. Sarkar, "New collision attacks against up to 24-step SHA-2," *Progress in Cryptology – Indocrypt 2008, LNCS 5365*, D. Roy Chowdhury, V. Rijmen, and A. Das, Eds., Springer, Heidelberg, 2008, pp. 91–103.
89. Y. Sasaki, K. Aoki, "Finding preimages in full MD5 faster than exhaustive search," *Advances in Cryptology, Proceedings Eurocrypt'09, LNCS 5479*, A. Joux, Ed., Springer, Heidelberg, 2009, pp. 134–152.
90. A. Shamir, E. Tromer, Factoring large numbers with the TWIRL device *Advances in Cryptology, Proceedings Crypto'03, LNCS 2729*, D. Boneh, Ed., Springer, Heidelberg, 2003, pp. 1–26.
91. V. Shoup, OAEP reconsidered, *Advances in Cryptology, Proceedings Crypto'01, LNCS 2139*, J. Kilian, Ed., Springer, Heidelberg, 2001, pp. 239–259.
92. *Contemporary Cryptology: The Science of Information Integrity,* G.J. Simmons, Ed. (IEEE Press, 1991).
93. A. Sotirov, M. Stevens, J. Appelbaum, A.K. Lenstra, D. Molnar, D.A. Osvik, B. de Weger, "Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate," *Advances in Cryptology, Proceedings Crypto'09, LNCS 5677*, S. Halevi, Ed., Springer, Heidelberg, 2009, pp. 55–69.
94. Y. Tsunoo, T. Saito, T. Suzaki, M. Shigeri, H. Miyauchi, Cryptanalysis of DES implemented on computers with cache, *C Cryptographic Hardware and Embedded Systems, CHES 2003, LNCS 2779*, C.D. Walter, Ç.K. Koç, and C. Paar, Eds., Springer, Heidelberg, 2003, pp. 62–76.
95. P.C. van Oorschot, M.J. Wiener, "A Known Plaintext Attack on Two-Key Triple Encryption," *Advances in Cryptology, Proceedings Eurocrypt'90, LNCS 473*, I.B. Damgård, Ed., Springer, Heidelberg, 1991, pp. 318–325.
96. P.C. van Oorschot, M. Wiener, Parallel collision search with cryptanalytic applications, *J. Cryptology*, **12**(1) 1–28 (1999)

97. X. Wang, Y.L. Lin, H. Yu, Finding collisions in the ful SHA-1, *Advances in Cryptology, Proceedings Crypto'05, LNCS 3621*, V. Shoup, Ed., Springer, Heidelberg, 2005, pp. 17–36.
98. X. Wang, H. Yu, How to break MD5 and other hash functions, *Advances in Cryptology, Proceedings Eurocrypt'05, LNCS 3494*, R. Cramer, Ed., Springer, Heidelberg, 2005, pp. 19–35.
99. X. Wang, H. Yu, W. Wang, H. Zhang, T. Zhan, "Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC," *Advances in Cryptology, Proceedings Eurocrypt'09, LNCS 5479*, A. Joux, Ed., Springer, Heidelberg, 2009, pp. 121–133.
100. D. Watanabe, S. Furuya, H. Yoshida, K. Takaragi, B. Preneel, A new keystream generator MUGI, *Fast Software Encryption, LNCS 2365*, J. Daemen, V. Rijmen, Eds. (Springer, Heidelberg, 2002) 179–194.
101. M.N. Wegman, J.L. Carter, New hash functions and their use in authentication and set equality, *Journal of Computer and System Sciences*, Vol. 22, No. 3, 1981, pp. 265–279.
102. D. Whiting, R. Housley, N. Ferguson, "Counter with CBC-MAC (CCM)," RFC 3610, September 2003.
103. M.J. Wiener, Efficient DES key search, *Presented at the Rump Session of Crypto'93.* Reprinted in *Practical Cryptography for Data Internetworks,* W. Stallings, Ed., IEEE Computer Society, 1996, pp. 31–79.
104. H. Yu, G. Wang, G. Zhang, X. Wang, The Second-Preimage Attack on MD4, *Cryptology and Network Security, CANS 2005, LNCS 3810,* Y. Desmedt, H. Wang, Y. Mu, Y. Li, Eds., Springer, Heidelberg, 2005, pp. 1–12.