Public Key Cryptography's Impact on Society: How Diffie and Hellman Changed the World

Paul C. van Oorschot*

Abstract. In 1975 and 1976, Whitfield Diffie and Martin Hellman conceived and introduced fundamental new methods that changed how communications are secured. Their landmark paper "New Directions in Cryptography" explained both *public key cryptography* and what would become known as *Diffie-Hellman key exchange*. These ideas, influenced and augmented by a few souls within a small community, set the world on a new course by establishing novel cryptographic techniques for protecting information transmitted over untrusted channels. Our aim herein is to consider how public key cryptography has changed the world, and in particular its impact on society. We review the original contributions of Diffie and Hellman, and provide context to relate these to pre-existing and subsequent cryptographic techniques. Aided by this understanding, we connect their contributions to resulting major changes in society. To retain accessibility for non-specialists, our treatment largely avoids mathematical details, while selectively introducing technical terms to maintain technical accuracy.

1 Security background

We begin with some basic concepts and terminology to develop a working vocabulary. When information is transmitted over a physical channel (*physical line*) such as a traditional phone line, cable, or optical fibre, the line may be physically shielded or isolated, to reduce the risk of unauthorized access such as by a physical wiretap. If such a communication channel is accessible to unintended parties, it is called an *open* or *untrusted channel*. In general, ordinary information (*plaintext*) sent over untrusted channels is at risk of interception. For example, plaintext sent over a radio channel is accessible to anyone with a suitable wireless receiver.

A common defense is to convert plaintext characters into a related sequence of characters (*ciphertext*) that are not meaningful even if intercepted. To do so, at the sender's end a sequence of instructions (called an *encryption algorithm*) is used to convert plaintext to ciphertext, which is then transmitted. To recover the plaintext, the operation is reversed at the receiver's end by a *decryption algorithm*.

In this way, encryption provides a *confidentiality* property, whereby the meaningful content is available only to authorized parties. Unauthorized parties cannot recover the plaintext because the encryption and decryption algorithms require a secret number, which may be viewed as a random string of 0s and 1s; 128 of these would be called a 128-bit *cryptographic key*. The aim is that only the sender and recipient (i.e., their computing devices) share this secret key.[1] Historically, decryption requires the same key as used for encryption; in this case we use the terms *symmetric-key algorithms* and *symmetric keys*.

Distinct from confidentiality or secrecy is the concept of *authentication*. The ability to recognize individuals (*entity authentication*) is taken for granted in human-to-human interactions, but more challenging in written communications. (Can you be sure that the postcard you have received is legitimately from your sister?) To provide a modest degree of authentication beyond relying solely on context and semantic content, we conventionally

*14 April 2020. A version of this paper is to appear in a forthcoming ACM book.

[1] Viewing the encryption process as an algorithm, the key is a required *input parameter*. For a given plaintext message, a different key results in a different ciphertext.

end handwritten letters with personal signatures. Historically, various alternatives have been used to provide arguably stronger assurances. One example is wax seals created using recognizable *signet rings*, in some cases affixed alongside a handwritten signature, and in other cases on an exterior envelope, thereby providing also a signal in case the envelope has been opened and/or the content altered. Such means serve to prevent or detect alterations, also called *integrity* violations. Mechanisms that convey assurances regarding the source of transmitted content are said to provide *data origin authentication*; in practice, such mechanisms also provide integrity protection, i.e., are able to detect if content has been altered. In dealing with computer data, it is helpful to define *data authentication* as the combination of data origin authentication and data integrity. We then try to avoid the ambiguous term *authentication*, instead using *entity authentication* or *data authentication*.

2 Context: motivation and environment

The invention of public key cryptography resulted from a combination of elements. These included the inventors' personal perspectives and motivation, and an environment that led to key management and data authentication being recognized as important open problems.

The need for digital signatures was a seed planted in 1970. While working on artificial intelligence research for John McCarthy at Stanford, Diffie was exposed to McCarthy's early interest in "*buying and selling through home terminals*" [68]. What we now call electronic commerce (*e-commerce*) was a new idea in this age before personal computers, when even home-based terminal connections to workplaces were rare. This led Diffie to recognize an open question [18, 88]: How would one authenticate digital transactions with remote parties, in lieu of assurances provided by handwritten signatures on paper contracts? Known techniques [18] for authentication at that time, using a symmetric key shared between buyer and seller would fail if one party originated a message but then denied this, claiming that the other party had done so. Indeed, parties in business and consumer transactions often have competing interests, while business partners on one project may compete on the next.

Moving now to encrypted communication between two remote parties, this conventionally required that a symmetric key be shared between them, prior to the beginning of the secure communication. The key is not sent over the same channel as the message—if that channel itself provided confidentiality, encryption would not be needed for the message. Thus conventionally, it was necessary to (1) know, ahead of time, who you planned to communicate with securely; and (2) pre-distribute a secret key to them over a trusted channel—thus the stereotypical briefcase chained to a courier's wrist. This is of course costly in both time and money. However in specific environments, such as the military and some government organizations, it is feasible due to, e.g., centralized organization with hierarchical reporting trees, and relatively static, well-known chains of command. Nevertheless, such pre-distribution of keys is particularly challenging at scale; according to Diffie, at one time the US National Security Agency (NSA) reportedly had "*the biggest printing press in the world at Fort Meade*" [17], used at least in part for printing keying material.

In May 1973, the US government's National Bureau of Standards (NBS, the precursor of NIST) solicited proposals, for "*algorithms for the encryption of computer data to ensure their protection during transmission over exposed communications facilities or while recorded on media in transport or in storage*" [60]. Following a second call in August 1974 [61], IBM submitted an algorithm that would become the Data Encryption Standard (DES), after its formal proposal by NBS in March 1975 [62]. In November 1976, DES was approved as a Federal Information Processing Standard (FIPS), FIPS 46.

This DES-related activity opened, for the first time, the possibility of broad use of encryption by the general public. In parallel, Hellman and Diffie, who had met in September 1974 [68], recognized key management as a barrier [18] to broad adoption of encrypted communications. They noted: if there are n users, for each to share a distinct key with each of $n-1$ others would require $n(n-1)/2$ or roughly n^2 keys across the system. This number grows quickly as n increases. Pre-distribution of this many keys, e.g., by physical

courier, would be impractical across the general public—due to both the number of users, and absence of the simplifying structures noted above.

As a separate issue, centralized key distribution was not an appealing solution for individuals who preferred not to rely on, or place trust in, governments and large organizations. Such individuals included Diffie, "*who grew up in a libertarian and leftist tradition*" [84]. In Diffie's words, "*I had a very counter-cultural viewpoint and it was very anti-central authority*" [88]. His interest in key management dated back to 1965 when he mistakenly[2] came to believe that for within-facility calls, NSA phone lines were encrypted, using keys distributed by a central facility [88]. He didn't see the point of such an architecture—his view was that cryptography should reduce the number of parties that one had to trust. Thus two communicating parties should use keying material known to them alone. A key management solution that required trusting a central facility missed the mark, leaving key management as an open problem in his mind.

In the fall of 1974, Diffie also first learned of the NBS plan to standardize encryption from Butler Lampson [88]. Believing that the US government might not standardize an algorithm unless they themselves could break it, Diffie began to ponder whether somehow a "trap-door" might be built into the algorithm, such that a party with special knowledge of the trap-door (but no one else) could easily break the algorithm—an idea that he credits [88] for inspiring public key cryptography. Thus the DES activity may be seen to have motivated the inventive contributions of Diffie and Hellman both through its heightening the need for key management, and by triggering the idea of trap-door cryptosystems.

As we explain next, public key encryption, and Diffie-Hellman key exchange, provided two different solutions to the key management problem. Public key cryptography also provided a solution to the open problem of digital-world signatures.

3 Inventive contributions

We are now in a position to list, and discuss further below, the primary contributions of Diffie and Hellman [18, 19]:

1. the concept of public key cryptography (*asymmetric* cryptography) for encryption;
2. the concept of digital signatures based on public key cryptography; and
3. two key management techniques for establishing a shared symmetric key between two parties—Diffie-Hellman key exchange, and by transferring a key from one party to the other after using public key encryption on it.

Diffie and Hellman also played a foundational role in disseminating knowledge on cryptography in general [20]. This was important as prior to their papers, scant technical literature on cryptography was publicly available. We now consider, in turn, the three items above.

Public key cryptography introduced a new type of encryption-decryption system. Here, instead of using traditional symmetric keys, each user now has their own *pair* of keys (E, D). E is called a *public key*, and D is called a *private key*. For encryption, the sender uses the public key (of the intended recipient) to parametrize the encryption algorithm. Only the intended recipient of an encrypted message has the corresponding decryption private key necessary for the decryption algorithm to recover the plaintext. Even the sender cannot reverse the operation to produce the plaintext from the ciphertext itself, since a user's private key is shared with no one else. This can be likened to "*a letter box or a night deposit box. [Any]body can put a message in and only one person can get a message out*" [16]. An essential property of such public key systems is that knowledge of the public key must not allow computation of the private key. Given this, each party's public key need not be hidden, and indeed, can be published, e.g., in a public directory for others to look up. However, while

[2]Diffie later came to know that beyond a regular phone for non-secret calls, the internal-use second phone of NSA employees only had physically shielded cables at the time.

the public key need not be kept secret, its data authenticity and integrity must be protected, lest an adversary substitute its public key for that of a legitimate intended recipient.

The second item introduced the concept of a *digital signature* based on public key cryptography. This uses a public key system, as just described, to provide data authentication. However, the public key system is now used in a different manner—changing both the order in which the public-private key pair is used, and which party's key pair is used (sender vs. recipient). To digitally sign a message, the sender uses their *own, private* key in the private-key algorithm (that algorithm is now viewed as a *signature* algorithm, rather than an encryption-decryption algorithm). The output is an encoded message, which conveys data authentication as follows. The recipient takes the encoded message, and provides the *sender's* public key to the public key algorithm (which is now viewed as a *signature verification* algorithm). If the output is an intelligible message, then this provides evidence that the owner of that public key originated the message—based on the assumption that only that owner should know the corresponding private key needed to create the encoded message.[3] An important point is that the digital signature (encoded message) depends on, i.e., is a function of, both the plaintext and private key. Thus for a given signature private key, in practice each plaintext message has a different digital signature; if a single bit of the plaintext is changed, the signature changes. This provides integrity protection, and eliminates from the digital world, the physical world problem of cutting and pasting a signature to a different message.

The third contribution addressed *key management*, giving two means to establish a (symmetric, secret) shared key K between two parties:

i) Diffie-Hellman key exchange (DH). This provides a specific solution to the *key distribution problem*: How can two parties remote from each other establish a shared key K by exchanging messages over an untrusted channel, such that no eavesdropper can derive the same key. The specific instantiation of DH given used *modular exponentiation*.[4]

ii) *key transfer* based on public key encryption. Here, two parties establish the shared key K as follows. The sending party chooses a random number for the symmetric key K, encrypts it using the encryption public key of the intended recipient, and sends the ciphertext (the encrypted symmetric key). The recipient recovers K by using their own decryption private key.[5]

Diffie originally thought that fast public key encryption instantiations would be found, entirely eliminating the need for symmetric key encryption; this is no longer expected [15]. Aside from encryption of short messages, the primary use of public key encryption is now key management, e.g., to encrypt symmetric keys to be used with fast symmetric-key algorithms that encrypt large messages.

Regarding specific contributions, Diffie and Hellman view their contributions as joint work resulting from a solid period of collaboration, while also agreeing [88, 35] that Diffie conceived (in May 1975) the idea of public key cryptography and digital signatures based thereon, while Hellman discovered (in May 1976) Diffie-Hellman key exchange. The ideas of public key cryptography and digital signatures appeared in a draft submitted December 1975 to the National Computer Conference [18]; Diffie included Diffie-Hellman key exchange in the June 1976 conference talk itself, as did Hellman in a talk in Ronneby, Sweden at the IEEE International Symposium on Information Theory, 21–24 June 1976. Their definitive "New Directions" paper [19] was sent to *IEEE Transactions on Information Theory* in June

[3] These ideas would be refined later by others, to address technical issues including what constitutes an "intelligible message", and to distinguish the class of signatures originally implied (later called *digital signature with message recovery*) from a second class that would turn out to be more practical (*digital signature with appendix* [37, 55], using cryptographic hash functions). These details are tangential.

[4] For technical reasons, this modular exponentiation used a large *prime number* p (this means that no integers divide p except 1 and p itself). It is implemented by sequences of squaring and multiplying integers, at each step keeping as the result the division remainder after dividing the interim result by p.

[5] Such key transfer relies on the sending party knowing the authentic public key of the receiving party; and because this uses public key cryptography, it was not implementable until later when RSA, the first concrete instantiation of public key cryptography, became known.

1976 and appeared as a pre-invited paper in the November 1976 issue, which physically arrived [89] in January 1977.

4 Supporting and related developments

While Diffie-Hellman key exchange provided a concrete solution for the key distribution problem, the above-mentioned 1976 papers offered no instantiations for public key systems or digital signatures—only "suggestive examples" conveying the desired properties. A concrete instantiation of a public key system, that could furthermore provide digital signatures as envisioned, was created in April 1977 by Ron Rivest, Adi Shamir and Len Adleman. This *RSA algorithm* became widely known through Martin Gardner's August 1977 column [28] in *Scientific American*, and a formal journal paper [72] published in February 1978.

We noted the need to protect the data authenticity and integrity of public keys stored, e.g., in a public directory. A means to do so was proposed in 1978 by MIT undergraduate Loren Kohnfelder [44]: *public key certificates*. These create a verifiable tie between a public key and an identifier asserted to represent its owner. Kohnfelder's certificates have evolved into today's widely used *X.509 version 3 public key certificates* [12].

Public key certificates brought a new requirement: verifying the binding between a public key and an owner requires trust in the public key of a third party, the *certification authority* (CA). One approach to address this, used in web browsers, is to embed CA public keys into browser software. CAs must now be trusted not to issue false certificates maliciously or by mistake, and browser vendors must be trusted on which CA public keys they embed—both counter to Diffie's original goal of reducing the number of parties that one must trust.

Before the ideas of public key cryptography or DH key exchange were conceived or published, a UC Berkeley undergrad, Ralph Merkle, began considering a challenging question: Using a communication channel on which an opponent could also listen, can two parties exchange messages to establish a shared key that the opponent cannot also learn? As already mentioned, the reason that couriers are used to pre-distribute keys over a separate channel is that sending a secret over the main channel exposes it to the opponent. Merkle wanted to use the main channel nonetheless, but in a way such that the opponent can *not* deduce the key—in other words, to establish *secure communications over insecure channels*. This phrase would be the title of his long-delayed paper [56], which in final form, succinctly detailed *Merkle Puzzles*. His earliest ideas on this were part of a project proposal for a Berkeley course in the fall term of 1974. The proposal as submitted was not well received by the instructor, and Merkle dropped the course, but continued working on this—the same problem Diffie and Hellman would call the *key distribution problem*, and find their own solution involving modular exponentiation. In early February 1976, Merkle came across a draft of their NCC paper [18] on public key cryptography—DH key exchange [19] was later—and contacted Hellman [85].[6] As elaborated next, Merkle's "*formulation of the problem*" [68] differed from using public key encryption to transfer a secret. Diffie notes: "*we took what Merkle had done, and produced a solution to it in a different way*" [16].

Merkle Puzzles involve neither digital signatures nor a public key encryption system per the NCC paper. Instead, one party sends the other a large set of "puzzles". Each consists of two items encrypted under a unique (symmetric) random key: a puzzle ID and a symmetric key (*puzzle key*). The second party chooses one puzzle and "solves it" by exhaustively trying keys; the encryption algorithm's key space is sized such that "solving" one puzzle takes moderate but not infeasible effort. The recovered puzzle ID is sent back to the first party. It is meaningful to the first party, who retains all the plaintext puzzles. Now both parties have a shared secret, the puzzle key, while by design an eavesdropper must solve many puzzles (not just one) to learn that secret. Merkle Puzzles are thus a "*demonstration that you could have a method where [...] we could force an eavesdropper to put in more work than we put in*" [69].

[6] According to Levy [50], Merkle's interest in the key distribution problem was mentioned earlier to Diffie by Peter Blatman, a mutual acquaintance, prior to the May 1975 conception of public key cryptography.

In summary, Merkle Puzzles allow two parties to establish a shared key by exchanging messages over a public channel. The second party recovers the key by using (part of) the received message, while the first uses information returned by the second plus (unshared) information retained after creating the puzzles. In comparison, DH key exchange is similar at one level, in that each participant sends the other a number or *public element* [88],[7] but in DH each derives a common symmetric key by combining the element received, with secret information that was (retained after being) used to generate the element sent to the other. DH elements are not cryptographic keys in the encryption sense, but being sent over a public channel, may be viewed as (key agreement) public keys used to form a shared key.

This explains how using "DH public keys" differs conceptually from using (e.g., RSA) encryption public keys for key transfer. Either DH-type or RSA-type public keys can be used to establish shared secrets. Calling DH key exchange a means for *public key distribution* rather than a *public key system* created confusion as to whether DH key exchange was itself a form of public key cryptography. Diffie noted [88] that had DH key exchange originally been called a "public key system", there may have been greater commercial value in two patents arising from Hellman, Diffie and Merkle: the DH key exchange patent [36], and a second with broader claims on public key encryption and digital signatures [37].

The paper on Merkle Puzzles [56] was originally submitted in August 1975, first rejected in a letter dated 25 October 1975, then revised and further rejected numerous times. Its final acceptance yielded an April 1978 publication date—at which point both public key cryptography and Diffie-Hellman key exchange were understood by experts. In the interim, Merkle became Hellman's PhD student. During this time [57] and after, Merkle made a number of groundbreaking contributions to cryptography. In recognition, Hellman suggests [35] that Diffie-Hellman be called Diffie-Hellman-Merkle key exchange.

The timeline of Fig. 1 (next page) summarizes the relationships between the above events.

In 1997, the UK government disclosed the work of three GCHQ employees: James Ellis, Clifford Cocks and Malcolm Williamson. Previously classified internal reports were released, as well as a summary paper [27]. This led to the following consensus.

1. Ellis recognized the possibility of public key encryption, calling it *non-secret encryption* (report dated: January 1970). Public key signatures were not envisaged.

2. Cocks is credited with finding an RSA instantiation (report dated: November 1973). This was for public key encryption (again, digital signatures were not suggested).

3. Williamson is credited with discovering DH key exchange (report dated: August 1976).

Ellis noted that Williamson's August 1976 report was written "*much later than he thought of it*" [27]. For reference, Diffie presented DH in his June 1976 NCC talk [18], though not in the written paper. Circa 1980-1981, some details of the GCHQ work became known to Diffie, who subsequently met with Ellis "*many times*" [16]. As Diffie has observed [16, 88], Ellis' original 1970 report itself provides no indication of motivation for public key cryptography, nor its potential use for key management; thus we lack evidence that this use was recognized.

In June 1984 Taher ElGamal, another Hellman-advised student, completed his PhD thesis [25]. Its final chapter and resulting paper [26] provided a new cornerstone for public key cryptography: concrete instantiations of both a public key encryption scheme and a public key signature scheme, each closely related to DH key exchange in their use of the same arithmetic structures. ElGamal public key encryption can be viewed as using DH key exchange to transfer a symmetric key to a second party. ElGamal's original signature scheme used modular exponentiation and a few further arithmetic operations involving the private key of the signing party. For these reasons, the ElGamal public key encryption and signature schemes are said to be *DH-based* (vs. RSA-based), but are distinct from DH key exchange itself, which offers a solution to the key distribution problem. The pair of ElGamal schemes provides an alternative to RSA public key encryption and RSA digital signatures.

A variant of ElGamal signatures called the *Digital Signature Algorithm* (DSA) was proposed in August 1991 by the US National Institute of Standards and Technology (NIST) as

[7] The number is called a *Diffie-Hellman exponential*, or in more recent terminology, a *key share*.

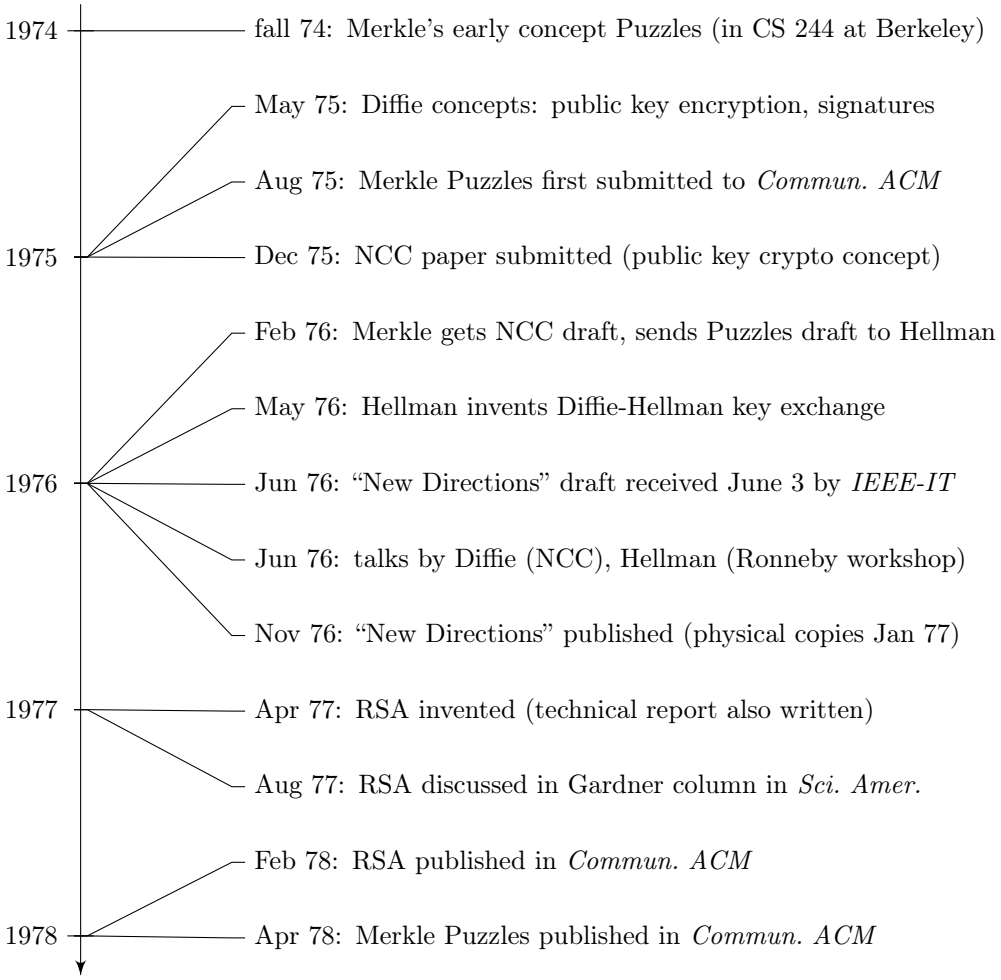

Figure 1: Timeline relating selected events in early public key cryptography.

a US standard (FIPS). This resulted from NSA influence [48], and an NSA algorithm attributed to David Kravitz [47], which they preferred over an RSA-based signature standard that might result in wider deployment of RSA encryption. This DSA variant of ElGamal was adopted as FIPS 186 in May 1994, and called the *Digital Signature Standard* (DSS).

Regarding DSS, an *elliptic curve* based implementation [41] was also standardized, first as a US banking industry standard ANSI X9.62. As further context here, in what has turned out to be a critically important advantage for ElGamal-based schemes, they can (unlike RSA) be adapted to use a variety of underlying mathematical structures (*finite cycle groups*). One of these in particular—the group of points on an elliptic curve over a finite field [55], as proposed independently in 1985 by Neal Koblitz and Victor Miller—provides major practical advantages including smaller key sizes for equivalent security, and fewer required computational steps (thus allowing faster implementations). The ElGamal schemes thus opened an evolution path unavailable to RSA, the most popular scheme over the first 30 years of public key cryptography. For this reason, while for many years he found RSA more appealing than DH-based systems,[8] Diffie now views RSA as "dead end" technology [88]. DH key exchange—which has long been viewed as, and continues to be, a preferred solution to the key distribution problem—is also amenable to elliptic curve implementations. Practical advantages are thus available from a suite of DH-based schemes with elliptic curve implementations: DH key exchange and variants of the ElGamal public key encryption and signature schemes.

5 Major impacts on society

Our discussion has explained the roles of Diffie and Hellman in the discovery of public key cryptography, and how the dissemination of their ideas led to developments by others. Here we selectively consider specific practical applications to convey a sampling of the enormous impact of public key cryptography on society. While cryptography, and security technologies in general, are often thought of in terms of providing defense and protection, here we will see that public key cryptography has had major impact also by enabling new types of activities and behaviors not previously considered to be feasible, or that appear unlikely to have otherwise emerged as rapidly or as successfully.

5.1 From early applications to secure messaging

The invention of public key cryptography had little noticeable impact on society over 1976-1985, the first decade after its discovery. The US government did show early interest: Sandia National Laboratories (Department of Energy), responsible for nuclear weapons command and control, began developing RSA hardware to support digital signatures in authentication applications [15]. One motivation here was a verification system for the US to monitor Russian compliance with a nuclear test-ban treaty [76].[9]

As telephones were the main mode of communication at this time, a natural target application was secure telephones. However, due to the significant computational requirements arising from the arithmetic operations in public key algorithms, time would be needed for technology to catch up with performance requirements. While the NSA's classified STU-III phones [15] would be one of the first large-scale uses of public key cryptography—initial estimates projected their number to be in the hundreds of thousands—their first deployment was not until the second half of the 1980s.

Over 1986-1995, public key cryptography attracted the attention of, but little evidence of broad adoption by, a conservative banking industry, which even in the early 1990s lacked

[8] As late as 2005, Diffie promoted RSA-based encryption and signatures over ElGamal-based alternatives, referring to RSA as "*[t]he one real solution, good solution that has been found for that approach ... it has become the most common public key system*" [16].

[9] Simmons has also detailed the challenges that arise in practical applications of authentication between mutually distrusting parties, and how public key cryptography provides a solution where symmetric key systems cannot, e.g., if one of these parties wishes to convince a third party that cheating has occurred [75].

the confidence in new cryptographic algorithms; prudence dictates that this emerges only over time. Bankers thus remained largely allergic to public key technology implemented in software alone. What did draw interest in this second decade was the maturation of *smart cards* or *chip cards*—integrated circuits with non-volatile storage (e.g., for cryptographic keys), but also processing capabilities. They were positioned as not just "*an improved traditional credit card*", but "*a multipurpose, tamper-resistant security device*" [33] with higher assurance than software alone (which might admit malicious software), and offering an additional factor for authentication.

Smart cards would go on to play an important role in the emerging mobile phone industry, led by the European GSM system with smart card SIMs (*subscriber identity modules*). However, GSM SIMs used only symmetric key algorithms, avoiding more computationally expensive public key algorithms. The inconvenience and risks of managing shared secrets (as necessary with symmetric algorithms) were ameliorated by a centralized architecture and strong cooperative agreements between telecom service providers. It would take longer for smart cards to broadly leverage the advantage that for authentication by digital signatures (including for personal identification, further below), a secret key need not be shared with the verifying party. Smart cards employing public key algorithms emerged later in the form of *chip-and-PIN* payment cards, again led by Europe here with EMV cards from the Europay MasterCard Visa consortium [53]; these have largely replaced magnetic stripe credit cards in many countries today. Other uses of smart cards include, e.g., in satellite pay television (*conditional access* systems). According to the International Card Manufacturers Association (ICMA) [40], 5.5 billion SIM cards and 5.6 billion financial cards (including, e.g., credit and debit cards) were produced worldwide in 2019.

During this second decade (1986-1995), a new focus for widescale deployment of public key cryptography was secure email. An early major commercial deployment was provided through IBM/Lotus Notes, a "groupware" product whose 1989 release included support for RSA encryption and digital signatures [13]. According to a 2005 article, "*The Lotus [Notes] public key infrastructure [PKI] is perhaps the most widely deployed PKI in the world, with more than 114 million licenses*" [95]. Larger deployments in other application areas would emerge (below). Ray Ozzie, a creator of Lotus Notes, is clear on the role of public key cryptography (PKC): "*There wouldn't have been a Lotus Notes without PKC*" [67].

Over 1990–1995, Privacy Enhanced Mail (PEM) [43] arose as the first major non-proprietary secure email effort. It would soon be superseded by S/MIME [24], which continues to enjoy support of major software vendors today. In parallel, PGP (Pretty Good Privacy) [94] was developed in the US by Philip Zimmermann, as the first of a sequence of products, companies, standards and services using the same PGP acronym for a variety of (not necessarily interoperable) file encryption utilities and secure email tools, suitable for small groups of technically skilled users. While easy-to-use secure email software such as Lotus Notes and competitors do exist and continue in use within closed communities (e.g., government departments and enterprise organizations), no widely interoperable secure email system is in sight [65]. Unsurprisingly, one culprit is key management, in this case, how to bridge trust in public keys not only across a few communities, but across essentially all email users in the world (as global interoperability is the default expectation for email). This is not a failure of public key cryptography—indeed, since managing and attaining trust is not a solved problem in the physical world, it appears naive to expect this from any technology alone, or that it will be simpler in the Internet's virtual world of remote parties who bear little or no accountability to each other, perhaps in distinct legal jurisdictions. The issue of managing trust across different administrative domains remains one of the biggest open problems in distributed security today [16], and PKI—which reintroduces centralization as part of its solution—is at best a partial solution .

In contrast to continued failure in the pursuit of widely interoperable secure email is the enormous success of public key cryptography in today's *secure messaging* world. In April 2016, Facebook's *WhatsApp*—a messaging application reported to have between one and two billion users—instantly began providing end-to-end signing and encryption of messages between users, using the *Signal* protocol [11] supported by public keys and an elliptic curve

based implementation [87]. The speed of deployment was possible due to an environment and architecture entirely different than secure email: WhatsApp enjoys the circumstance of a single administrative domain in a position to ensure all users access to the public keys of all others, and its encryption functionality did not require any extra action by users for installation, configuration, or actual use. This is a grand success of public key cryptography in the second decade of the 21st century. However, despite the numbers, compared to other uses of public key cryptography (below), no significant social impact is yet apparent [14].

5.2 TLS and securing browser-server communications

In 1995, twenty years after the discovery of public key cryptography, its impact on society remained relatively small, or at least largely unnoticed. This changed in March 1995 [70], when the *Secure Sockets Layer* (SSL) was first deployed in a Netscape browser, Navigator 1.1. SSL and its successor, the *Transport Layer Security* (TLS) protocol [71], are recognized as the key technologies that enabled the rise of e-commerce. From its beginning, a design goal of SSL was to support the use of credit cards over the Internet—enabling convenient online purchase of goods and services by new users with an existing card.

To provide some context about TLS (which hereafter refers to TLS and/or SSL), we begin with HTTP (the HyperText Transfer Protocol), the basic application protocol used for exchanging information between a client (browser) and a web server. HTTP transfers information as plaintext. To instead send the same data securely, one can think of first setting up a "secure tunnel". The sender puts the plaintext (HTTP data) into one end of the tunnel, and the receiver extracts the plaintext (HTTP data) at the other end; the idea is that the tunnel represents the data being in a protected state (encrypted). TLS essentially provides the tunnel. Putting HTTP data into the TLS tunnel is referred to as "sending HTTP over TLS", with this combination denoted by *HTTPS*.

TLS provides a secure tunnel in the following sense. First it establishes a shared session key between the client and server (using key establishment via public key cryptography), and then that key is used to encrypt the data sent between client and server. Since TLS is designed to facilitate a client sending credit card information to the server, it is important that the client has some assurance of the server's identity, i.e., authentication of the server to the client. This is achieved by the server having a public key certificate that is sent to the client. The client software is preconfigured such that it can verify legitimate certificates. The public key in a verified certificate is then suitable for use in the key establishment stage, and the resulting shared symmetric key is called an *authenticated key*. This process thus involves two stages: a set-up phase for key establishment, followed by a communication session involving transmission of encrypted data. The 2018 update of TLS to version 1.3 [71] removed support for key establishment via RSA key transfer. This leaves key establishment via DH and its elliptic curve implementation (ECDH), respectively denoted DHE and ECDHE when each participant uses new DH exponentials each session.[10]

Further regarding TLS, a 2018 empirical study [46] indicates that use of RSA in TLS dropped dramatically in the second half of 2013, mirrored by a corresponding increase in ECDHE to roughly 90% of negotiated connections. This was strongly correlated in time with the "Snowden revelations" of June 2013, i.e., the intentional disclosure by Edward Snowden of classified documents revealing massive surveillance programs involving the NSA and international governments. This heightened awareness of public vulnerability to eavesdropping enabled by default protocols whereby possession or recovery of long-term secret keys allows decryption of encrypted traffic. This apparently triggered changes by browser providers, e.g., making ECDHE a default for TLS in browser sessions.

TLS is best known for its use associated with HTTPS as above, but was designed to also accommodate secure transfer of data by applications (protocols) other than browsers. Early example applications were link-by-link encryption of email between mail servers (mail

[10]This leaves TLS 1.3 supporting only key establishment protocols for which symmetric keys established, if deleted when a session ends, cannot be later recovered even if a long-term secret is compromised. The desirable resulting property is called *forward secrecy*. The "E" in DHE and ECDHE denotes *ephemeral*.

transfer agents); encryption between email clients (mail user agents) and their mail servers; and encryption between clients and servers exchanging files using the File Transfer Protocol (FTP). This accommodation plan has turned out to be wildly successful [3, 38], and TLS is now used as a general utility for secure data transfer by a wide variety of applications such as web- and video-conferencing, office productivity software, instant messaging, and applications connecting to cloud storage.

Viewing TLS as a utility providing a *secure session* for sending data from other applications or protocols helps explain how TLS provided the foundation on which to build secure online purchasing, online banking, and related e-commerce applications. The main problem that it solves is not so much in providing encryption and decryption—in fact, the design of cryptographic algorithms themselves turns out to be a well-solved problem—but rather, addressing key management for encrypted communications, often with brand new or very recent business partners, in many cases which have never been met in person. E-commerce itself is addressed separately below in Section 5.6.

5.3 Secure remote access: SSH and VPNs

TLS was possible due to the maturation of several technologies, including RSA and public key certificates, which themselves followed directly from the seminal concepts of Diffie and Hellman. This maturation enabled other new core tools and secure communication architectures that have been heavily leveraged by society-changing applications, two of which we now discuss. Both provide means to securely transmit session data between remote parties. Both follow the conventional two-stage secure communication process noted above for TLS: a session set-up phase (including key establishment to obtain authenticated symmetric keys) followed by the main session (transferring target data, suitably encrypted). First we mention the SSH protocol [91], and then VPN architectures [77].

SSH (Secure Shell) is a software protocol, typically bundled with a suite of applications, first released in 1995 by Tatu Ylönen—just as public key's third decade began. The basic protocol begins with a session set-up that uses public key cryptography, including certificates, to establish a session key between two parties called a *client*, which initiates the communication, and a *server*. As in TLS, an *encrypted tunnel* is provided. Plaintext data goes into the tunnel at one end, and is transformed into encrypted data that is sent over the communication channel; at the other end, it is decrypted and plaintext is recovered from the channel. In its simplest form, SSH is used to provide *secure remote access*, so that for example, from a client machine at home, you could log into an account on a server machine located at your workplace, with essentially the same security as if you were physically at your workplace and locally logged in. More generally, such an SSH tunnel can be used to transport (with encryption) data from various other applications that those original applications would otherwise send as plaintext from a source to a destination.

The term *virtual private network* (VPN) refers to a general architecture that provides an encrypted tunnel between communicating devices at distinct locations connected over public networks, analogous to SSH above but at a broader level—whereas SSH provides the secure tunnel for one, or a small number of specific applications, a VPN provides a "bigger" tunnel that secures most or all of the applications that transfer data between pairs of devices. This is useful, e.g., for securing all corporate data sent between two branch offices in different cities, or all of the traffic between a home user (who employs a large number of different applications) and their workplace. While more powerful, VPNs are generally more complicated to build and deploy than SSH, they involve more infrastructure (including more complex key management options [42]), and historically have required corporate-level operations support; however, simpler VPNs are emerging.[11]

Regarding impact, the ability to easily and securely connect to remote locations over public networks, and to securely use mainstream applications as if physically present at distant locations, has changed how people work. For example, it has enabled wholesale *telecommuting*, as well as part-time and evening access to workplace servers. Working from

[11] For example, support for *WireGuard* VPN technology [23] was built into the Linux kernel in 2020.

home is a major societal change. Using technologies like SSH and VPNs to allow two geographically distinct branch offices to operate as if a single co-located office, and to allow a company in San Francisco to hire an expert working remotely from Toronto, are likewise significant changes, now taken for granted. These changes did require other supporting advances, e.g., reliable communication networks with acceptably low latency and sufficient bandwidth. However, without key management enabled by public key cryptography, it is unlikely that ubiquitous remote access functionality would be with us today—as symmetric key management alternatives are more expensive, less convenient, and less automated.

5.4 Code signing, software update, and personal identification

Regarding the invention of public key cryptography, public key encryption is often headlined, but a strong case can be made for public key signatures as the unsung impact hero. Signatures play a critical role in countless protocols, including key establishment protocols such as authenticated Diffie-Hellman key exchange [21, 42], for assurance that resulting symmetric keys are shared with the intended party. As further examples showing that the elegant functionality of public key signatures is not easily replicated by symmetric algorithms, here we discuss two other high impact uses. This first is digital signatures on software, enabling major changes in how software is distributed. The second is digital signature use in personal identification applications with passports, national ID cards, and other physical tokens.

Consider updates to operating system software. Historically, updates were distributed via physical storage media (e.g., by CD ROM and earlier, floppy discs). Today, operating system updates are primarily downloaded over the network. Already in 2006, it was estimated that each patch released for Microsoft operating systems and distributed by the Windows Update automated update system reached 300 million users [31]. Digital signatures play a critical enabling role here: the software is digitally signed at the source, and the signature is verified by the operating system before installation. This protects against malicious software being substituted for the authentic code, and provides increased assurance of the origin of software, as well as its integrity (i.e., that it has not been altered since the time of signing).[12] Software signing (*code signing*) is now standard practice among major operating systems.

Software signing is now also common practice for application software (not just operating systems). This works as follows. The software source (*publisher*) obtains a public key certificate (*code signing certificate*) that identifies the publisher's signature public key. The certificate itself is signed by a third party (CA) such that the operating system can recognize a trusted signature verification public key. The operating system verifies both the certificate, and (using the public key from the certificate) the signature on the application software in question, before allowing that software to be installed. On some operating systems, software applications that are unsigned may still be installed after explicit approval by the user through a dedicated pop-up dialogue. Code signing requirements for core operating system components (e.g., *kernel modules* or *drivers*) may differ, or be more stringent, than for application-level software and other executables including third-party libraries.[13]

Software signing has enabled fundamental changes in how software is distributed. Software update frequency is now far greater, with greatly reduced roll out delays—including for major operating system upgrades, minor functional updates, and security fixes (*patches*). Upgrades can now reach major proportions of their target users within hours or days if desired, whereas in corporate environments in the 1990s, upgrades may have occurred at scheduled intervals of six to eighteen months, as central IT departments created new "disk images" with corporate-approved applications and operating system upgrades for internal distribution via physical media—manually visiting user workstations and servers one by one. As an important technical property of signatures here, software can be signed once and then verified by arbitrarily many devices using a single trusted public key, i.e., the sign-to-verify

[12] A valid signature gives no guarantees about software quality or safety, but prevents otherwise easy paths for installation of malicious software, and provides some traceability if problems are subsequently found.

[13] For example, on Windows using *Authenticode*, kernel drivers must be code-signed also by Microsoft [45].

relationship is one-to-many. In contrast, standard symmetric key authentication techniques would require per-client data keys, at significantly greater cost and complexity.

By public key's fourth decade (2006-2015), the majority of mobile phone users were taking for granted the ability to instantly download and install new phone applications from their providers' application markets. These applications are digitally signed—e.g., by Apple for applications from its App Store, and by independent software developers in the case of Google's Play Store for Android.[14] Similarly, commercial desktop operating systems (e.g., Windows, macOS) now commonly look for verifiable digital signatures before installing new applications downloaded from the Internet. Cornerstone desktop applications, such as major web browsers (e.g., Google Chrome, Mozilla Firefox) and productivity software (e.g., Microsoft Office suite, Adobe Acrobat) typically enable software auto-updating by default, or heavily encourage users to turn this feature on, in order to propagate security updates as quickly as possible, bearing in mind that manual triggering of upgrades by many users is slow to occur, if at all. While other major technology advances were again essential—including data communications bandwidth/transmission speeds, and software update architectures—digital signatures have facilitated online updates while limiting new security exposures.

Distinct from software authentication, personal identification is another application category where the functional advantages of public key technology are now widely leveraged in practice. Specific examples include integrated circuit (IC) contactless chips within national ID cards and passports, and other physical tokens similarly capable of generating digital signatures. As one example, Belgian eID (electronic ID) cards can generate digital signatures valid for legal contracts and declarations to the government [53].[15] Passports with embedded smart card chips, called *biometric passports* (*e-passports*), were already common in the 2000s, and are now available from most countries worldwide. As standardized by International Civil Aviation Organization (ICAO) Document 9303 [83], a portion of the passport information (including data representing a facial image, and optionally other biometric data) is machine readable and digitally signed; a passport reader verifies this static signature for data authenticity assurance. Using the *Active Authentication* option, an RSA private key in the chip can also be used to sign a random challenge number; passport reader verification of this dynamic signature response provides authenticity assurance of the physical passport.

Further regarding personal identification, a broad industry consortium called the FIDO Alliance has standardized interfaces and protocols for use of physical tokens capable of generating digital signatures to improve the security and usability of user authentication. The tokens, with early versions promoted by Google in 2016 as *Security Keys* [49], contain private keys associated with a user. A token generates signed responses to authentication challenges from web sites with whom the user has previously registered the token (with per-site public-private key pairs). A collection of FIDO specifications includes those governing use of such technology to either augment or entirely avoid use of passwords.[16]

5.5 Personal privacy, Tor and Bitcoin

Public key cryptography has also enabled "privacy friendly" protocols and tools that are fundamental to larger technology efforts and services—many centered on the importance of anonymity, and freedom from both surveillance, and censorship of information and use of web-based services. Examples are sites facilitating whistleblowing, tips to police, and tools to counteract the abuse of personal information by authoritarian regimes. Related also are technologies for which privacy is considered a desirable feature, such as digital cash. Here we discuss two prominent examples: a privacy service called Tor, and the Bitcoin cryptocurrency system.

We first mention the general *traffic analysis problem*. It asks: How can one protect not only the plaintext content of a message, but also hide related *metadata* such as the

[14] In the Google Android signing scheme, a currently installed software application itself specifies which signatures will be recognized as valid for future updates to that same app [90].

[15] These use 2048-bit RSA as of 2014.

[16] See, e.g., FIDO U2F/Universal Second Factor [79] and UAF/Universal Authentication Framework [51].

identities of the participants in a communication, and the time of the communication?[17] David Chaum considered one aspect of traffic analysis in a February 1981 technical note [6], namely *sender anonymity*—how to conceal the identity of the source of an email message. His solution involved a relay node called a *mix* node (because it also mixed in traffic from multiple sources), and allowed a response to the anonymous sender. A variation involved iterated use of public key encryption to "wrap" a message in encryption layers. The layers were peeled off sequentially by a series (*cascade*) of such mix nodes.

Such networks, later called *mix-nets*, would inspire an anonymous communication service publicly launched in 2003. Called Tor [22] for "The onion router", it improved on a preliminary design from 1990s work supported by the US Naval Research Lab involving Paul Syverson, who bridged both design groups. Tor is commonly used by first installing a customized browser, the Tor Browser, which accesses the Tor network to provide an encrypted connection to a target site through a varying set of three Tor servers (relying in part on TLS). This facilitates anonymous connections to any services hosted on the target site. Regular web sites can be directly accessed via Tor, while other services have been customized to run using the Tor network. Thus for example, Tor may be used in conjunction with SSH, peer-to-peer clients, email, and to access Facebook. Among other uses, Tor is promoted as providing privacy from advertisers and as an anti-censorship tool, allowing anonymous access to online services that a government might otherwise block or surveil.

Tor also supports what are called *onion services* (introduced as *hidden services* in 2004), which provide so-called *responder anonymity* to a service or web site. Thus a server offering, say, instant messaging, a discussion forum, or anonymous blogging can provide this service while its (TCP/IP) network address remains hidden from both clients (users) and web search engines. The Tor network connects the client to the hidden service through servers called *rendezvous points*. This requires extra set-up by the service, and advertising details such as a public key needed to access the service, and from which an identifier to reach the service is generated, and made available to users via Tor's *onion services directory* (formerly *hidden services directory*). For broader context, the *deep web* consists of web pages inaccessible by standard search engines; Tor hidden services are a subset of the deep web, and compose the *Tor darknet*. The *dark web* is the set of all darknets, including those beyond Tor (with access to each darknet typically requiring specialized software).

Tor has been the subject of tremendous attention by a community of privacy researchers.[18] It also enjoys heavy real world use, with Tor use estimates ranging from 2 to 11 million distinct client IP addresses daily circa 2018 [52].[19] We highlight that here again, public key cryptography is essential to the design of a high-impact technology. Tor is also not without controversy—it can be used to hide illegal activities as discussed later, notwithstanding US Naval Research Lab (NRL) origins and funding support from several US government sources. We now turn our attention to cryptographic-based currencies (*cryptocurrencies*).

A long line of proposals aiming to create untraceable *electronic cash* [63] dates to the 1980s, including early work by Chaum and others [7]. The cryptocurrency revolution of the 2010s was triggered by the October 2008 appearance of a paper describing *Bitcoin*, a new digital currency system offering units of exchange called (lowercase) *bitcoin*. The importance of public key cryptography is succinctly conveyed in that paper, by its inventor Satoshi Nakamoto (a pseudonym) [58]:

> *We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.*

In digital currency systems, coins are trivial to duplicate (being digital strings), but trivial

[17] Traffic analysis of enemy radio transmissions during WW2 not only enabled British inferences about German activities, but also about German key management which, due to operator errors in use of keying material, allowed German ciphers to be broken [86].

[18] For a bibliography of papers on anonymity, including Tor, see: `https://freehaven.net/anonbib/`.

[19] Tor's goals of anonymity and hiding IP addresses complicate both exact use measurements, and determining what fraction of client use is due to human users versus automated software or even malware.

duplication of coins is intolerable. This gives rise to a well known issue: some means is needed to prevent a coin's owner from spending it twice (*double-spending*). The common approach in digital cash proposals involves relying on a trusted central authority to address this. In contrast, Bitcoin's novel design avoids this, by using a consensus-based scheme and what are called *blockchains* [63]. Regarding the details of blockchains (which are of independent interest beyond cryptocurrencies), for our purposes it suffices to note that they protect the integrity of a public record of transactions (*distributed ledger*), and that this transaction data includes digital signatures. Public keys are also used to represent the owners of coins (per the above quote from Nakamoto). Thus public key cryptography is fundamental to Bitcoin, and other cryptocurrencies. As one measure of the impact of Bitcoin and related digital cash systems, the total value of the top 10 cryptocurrencies in circulation as of late August 2019 (at then-current market prices for respective cryptocurrencies) was reported to exceed USD240 billion [30].

5.6 Electronic commerce and the digital economy

We now move to general consideration of electronic commerce and the digital economy. Here, one observation we pursue is that the largest US companies by market capitalization are those heavily involved in e-commerce and cloud services.

Regarding the impact of public key cryptography on society, Diffie [88] has noted that "*very few things have as much visible impact on the world as this does*", clarifying that some higher-impact contributions in other areas of computer science are less visible "*because the individual algorithms didn't take hold in the world and become so well known*". Along the same lines, in their 2016 book Dorothie and Martin Hellman assert: "*Today, that technology secures your electronic banking and your Internet credit card purchases. It also secures $5 trillion a day in foreign exchange transactions*" [34, 59].

Commerce, whether entirely in the physical world or e-commerce, involves exchange of goods for payment. This requires trust by the buyer that the ordered goods will appear as expected, and by the seller that payment will be received. The turning point for e-commerce was the emergence, and wide adoption, of TLS (above). Initial hesitations to use the Internet for commerce, e.g., due to confidentiality-related concerns, have largely been eliminated by ubiquitous and transparent encryption—enabled by convenient key management. Authentication and authorization, as enabled by digital signatures, have arguably helped even more in establishing trust in, and trustworthiness of, the commercial Internet.

Dell Inc., the American computer company, had a strong beginning building custom-ordered computers for businesses. A complementary new dimension of growth emerged through online direct sales to retail individuals in the mid 1990s. Enabling factors included web browsers, and SSL allowing confidential Internet transit of credit card information. At the March 10, 2000 peak of the Nasdaq stock market before the (first) Internet bubble burst, Dell's market capitalization was $178 billion [9].

At that peak, other large capitalization Nasdaq stocks included: Microsoft ($713B), Cisco ($633B), Intel ($545B), and Sun Microsystems ($200B). Since then, Microsoft transformed from a desktop operating system (Windows) and productivity suite (Microsoft Office) company to one largely centered on the Internet. A number of its major products including its productivity suite and email services were moved to a hosted-online model (*software as a service*) after investing heavily in the Internet Explorer browser, and major investments were made in its Windows Azure cloud computing service. Microsoft currently remains among the top four Nasdaq stocks (see Table 1). In contrast, the large hardware-based companies of 2000 (Intel, Cisco, Dell, Sun) have faded in relative capitalization. Other non-Nasdaq capitalization leaders from 2000, such as the large multi-national oil companies have similarly fallen behind Internet companies such as Google, and Internet technology companies such as Apple. As Table 1 shows, today's largest US public companies are precisely those that dominate e-commerce and the digital economy.[20]

[20] Regarding the dates in Table 1, 7 Oct 2000 coincides with statistics in Standard & Poor's annual book [80], while 25 Feb 2020 is when data was tabulated for this article by the author.

	Market Cap. (25 Feb 2020)	Revenue (fiscal 2019)	Market Cap. (7 Oct 2000)
Apple (APPL)	$1305B	$260.2B	$7.2B
Microsoft (MSFT)	$1300B	$125.8B	$297.6B
Amazon (AMZN)	$1000B	$280.5B	$11.2B
Alphabet (GOOGL)	$ 976B	$161.9B	IPO: Aug 2004
Facebook (FB)	$ 572B	$ 70.7B	IPO: Feb 2012

Table 1: Five largest US stocks by market capitalization. Notation: $(USD), B(billion)

Google is sometimes called a "pure" Internet company, in that it relies on network connectivity and remote access to deliver essentially all its services. Encryption and/or authentication are critically important in, for example, email services (gmail), Google's Chrome browser, installation and update of Android apps downloaded from their Play Store, cloud storage services (Google Drive), and cloud-hosted productivity tools like Google Docs. In support of Google's stated plan to make HTTPS the default for browsing, and "*to achieve 100% encryption across our products and services*", the company hosts an ongoing Transparency Report on HTTPS [32]. It depicts adoption of HTTPS and encryption across its major products, e.g., for its Chrome browser, reporting (for each of five OS platforms: Windows, Android, Chrome OS, Linux, macOS) the percentage of HTTPS browsing time, and the percentage of pages loaded over HTTPS. The reported use of encryption by gmail and Google Drive is now stable at 100%.

Similarly illustrating broad HTTPS use, a statistics page[21] from the Let's Encrypt project reports that as of early 2020, over 90% of web pages loaded in the US by Mozilla's Firefox browser used HTTPS. The Let's Encrypt project itself has greatly aided web site deployment of HTTPS not only through free TLS server certificates, but also software tools that simplify their acquisition and management, thus removing two deployment obstacles.

Facebook is the most recent arrival among the dominant Internet players. It rules social networking and supports instant messaging through various products including Facebook Messenger; includes the photo- and video-sharing service Instagram (as of 2012); and now also WhatsApp Messenger (as of 2014, and discussed earlier), which supports instant messaging, voice and video calls. Facebook's single sign-on (SSO) system, which provides convenient access for users as they interact with non-Facebook sites, relies on TLS to protect user login data and access tokens in a distributed authentication and authorization infrastructure [81]. More generally, as a company whose services are network-delivered via remote access, Facebook relies heavily on public key cryptography, as do the other Internet giants.

Apple (the company) also relies on encryption and authentication, but unlike Google and Facebook, heavily leverages physical devices (e.g., iPhone, iPad, Mac computer) sold to consumers to sell related software and services. Apple users rely on HTTPS in either Apple's own Safari browser or others (on both iOS and macOS platforms), but its business model does not rely on, e.g., browser-generated advertising revenue as in Google's case. Apple's iOS App Store, Mac App Store, and iTunes store involve download and update of digitally signed applications, verified by the operating systems on the host hardware. With a few exceptions (e.g., downloaded vs. streamed music and movies), Apple services and applications rely on ubiquitous and essentially continuous connectivity. Apple's iCloud provides cloud storage for end-user music, photos, and documents, and cloud-based hosting of its own productivity suite (iWork). Apple's own mail client (Mail), which supports S/MIME, can be configured to work with various email providers, including its own (through iCloud). Apple's popular proprietary instant messaging app (iMessage) uses public key technology as expected. Many Apple products and services rely on public key cryptography, but perhaps less essentially than its peers and competitors, due to a centralized model and control of hardware devices.

Amazon (the company) was founded in 1994, just prior to the introduction of SSL. It

[21] `https://letsencrypt.org/stats/`

began selling books online; now it sells almost everything. Amazon established a market presence by allowing use of an existing credit card for payment over the Internet—as discussed, a specific task for which SSL was designed [70]. The rest is history, as they say. Online shopping has dramatically changed human behavior, and how people buy goods including food. Many major retailers with physical stores are now essentially required to offer complementary online purchasing options. Today, Amazon not only leads in e-commerce market share, but among other activities, is a major force in cloud computing services, through its Amazon Web Services (AWS) arm following the 2006 introduction of its Elastic Compute Cloud (EC2).

Cloud computing itself is a major technology service sector, with market revenue of over $131 billion reported in 2015 [39]. Because the services are remote from the client, it relies critically on cryptographic security—and with today's implementations, this means public key cryptography. Secure remote access of cloud services [78] is commonly accommodated via browser (TLS), SSH, and other programmatic interfaces beyond our present scope.

We end this section quoting Hugh Williams from a 2017 interview, with what appears to be a generally accepted view on public key cryptography [88]: "*These results became the cornerstone of Internet commerce.*"

5.7 Detrimental impacts and illicit activities

As we have seen, public key cryptography has hugely benefited society, as an enabling technology. Unfortunately like many tools, it can also be employed for activities that are illicit or otherwise detrimental to society. Here we note a few prominent examples, either involving human users directly or through malicious software (*malware*). While this issue deserves serious attention, including by researchers, easy answers are elusive, and hasty conclusions—such as the recurring suggestion to provide "backdoors" allowing decryption by law enforcement authorities—should be discouraged [1]. As one rough analogy from recent times, automobiles driven into crowds have been used to intentionally injure and kill innocent individuals—but few societies have outlawed automobiles.

Encryption of communications, including instant messaging and email—supported by public key cryptography's key management—is used by criminal elements of society to hide information about their online (and offline) activities. This of course complicates efforts by law enforcement and other authorities to understand and stop such activities. Browser-based tools such as Tor hide both traffic content and communications metadata (identities and networking addresses). For data transfer between client devices and servers involving HTTP, whereas cleartext HTTP traffic is easily monitored, encryption via HTTP over TLS (i.e., HTTPS) defeats basic monitoring. Thus for example, the broad support of TLS makes it a handy tool to hide information in transit to and from web sites (or between peers) hosting illicit content such as child pornography. TLS is used not just directly by human users, but also by malware. A detailed empirical study of uses of TLS, published in 2018 [2], reported that 10% of the malware samples in its dataset used TLS for network communications.

Encryption of locally stored files on computers and laptops impedes investigations and prosecution of criminals by preventing access to stored content (that would otherwise be stored as cleartext), even if computers are seized for evidence under legal search warrants. A typical use of public key cryptography with stored data is as follows: a password, entered by the user, is converted into a symmetric key, which is used to decrypt the stored private key of a public-private key pair. The private key is then used to decrypt per-file symmetric keys, which themselves are used for encrypting and decrypting local stored files. By this approach, the file encryption program (e.g., a utility preparing files for backup in the cloud) need not be trusted with a long-term master secret, using instead a public key.

Encryption is also used to hide the existence and details of ongoing computer intrusions. For example, if enterprise computers are compromised by remote agents (*outsiders*), or in cooperation with internal rogue employees (*insiders*), encryption facilitated by TLS or SSH can be used to conceal the extrusion of data from enterprise computers to external machines, or commands sent from the remote external agents to the compromised machines within an

enterprise. Standard network-based monitoring and content analysis is impeded or entirely defeated by such encryption.

The term *botnet* is used to describe a collection of compromised devices (thereafter called robot nodes or *bots*). Each bot runs unauthorized software under the network-based control of one or more remote agents called a *botmaster*, which communicates to the bot through one of various *command and control* (C&C) messaging structures. In advanced botnets, C&C communications (including also malware updates) are cryptographically protected, with C&C messages often digitally signed; this stops simple reactive measures that might otherwise easily disrupt the malware. As an example, the well known *Zeus* banking Trojan malware (circa 2013) used RSA for signing C&C messages [4]. For technical reasons [73], simpler symmetric-key based methods suffice for botnet encryption key management.

Ransomware involves the use of malicious software to extract ransom payments, which must be sent to a remote server. A common variation relies on public key cryptography as follows. Files on a victim's filesystem are encrypted by malware, with the promise (often honored) that in return for payment, decryption keys will be provided via remote network connection, allowing the files to be restored. An essential feature of public key cryptography here is that even if the ransomware is reverse engineered, a decryption key is not there to be found in the software (rather, only the ransomware public key is present). This ransomware idea, first published in 1996 [92], became reality twenty years later [93]. As one example, the 2017 *WannaCry* ransomware incident infected over 200,000 Windows-based computers across the world, e.g., shutting down computer systems at National Health Service hospitals in the UK among many other impacts. WannaCry creates a per-victim RSA public-private key pair as a core element of its technical design. Notably, ransomware payments are also commonly requested in bitcoin for anonymity reasons (untraceability is a design goal of most cryptocurrencies). Thus ransomware leverages public key cryptography several ways.

An empirical study published in 2019 [3] looking at beyond-browser uses of TLS, including illicit uses of Tor, noted a steady increase in Tor use in the study's historical database of malware samples. An early example of a Tor hidden service that attracted media attention was *Silk Road* [8], an anonymous online marketplace that operated between 2011 and 2013, largely selling controlled substances including both prescription and non-medicinal drugs; payments were settled in bitcoin, hindering traceability. Also specifically regarding hidden services accessible via Tor, a study based on data collected in 2013 [5] reported that 44% of the hidden services involved selling at least one of: drugs, adult content, fraud or counterfeit products (including stolen account credentials, credit card details, fake passports), or weapons. Other miscellaneous services included money laundering. The same report notes:

> *The number of hidden services with illegal content or devoted to illegal activities and the number of other hidden services (devoted to human rights, freedom of speech, anonymity, security, etc.) is almost the same [...] Statistics of the popularity of hidden services look more distressing, however. The most popular onion addresses are command and control centers of botnets and resources serving adult content. The Silk Road market place is among 20 most popular hidden services.*

Another paper exploring Tor hidden services, from 2016 [66], finds generally similar hidden service categories, also noting for example bitcoin services (allowing bitcoin acquisition using mainstream payment forms, and also laundering). A 2017 paper [29] reports finding 23,585 onion sites presenting hidden services.[22] There remains disagreement as to what extent Tor is used for illegal versus beneficent use, and also on the number of hidden services sites—a number difficult to measure accurately, given that the design intent is hidden, anonymous services (within some context of the meaning of those words).

Beyond hidden services, a broader general concern is growth of online underground economies [82]. These would be especially challenging to disrupt if run entirely on a currency of untraceable cash. An ecosystem out of view of traditional banking oversight re-

[22] While the analysis tools may have missed some sites, this number may also over-estimate the actual number, if what is counted is unique advertised addresses rather than distinct onion services.

moves the ability to track organized crime through regulation-driven monitoring of currency transactions. On the other hand, monetization typically involves use of conventional banking systems at some point. For example, conventional credit cards are relied on in many Internet-based schemes that extract payments from users on false premises (e.g., fake antivirus products, sales of counterfeit software and pharmaceuticals). Evidence suggests that many such networks can be effectively disrupted by disabling the conventional bank accounts used by a small number of merchants [54]. It should be noted also that many of these activities need not rely specifically on public key cryptography, and use conventional credit cards rather than cryptocurrencies.

The above is meant to give an illustrative set, rather than an exhaustive list, of negative impacts that may be attributed in some way to public key cryptography. Other examples exist, with a relative severity of impact that is often subjective, such as in the case of using VPNs to evade geographic-based content licensing—e.g., Netflix users in Europe desiring access to US-based Netflix programming might use an origin-hiding VPN or proxy service to access US servers. A main point is that public key cryptography, including that which facilitates encryption and anonymity, is available for use by all elements of society.

6 Concluding remarks

In 1988, Diffie wrote about "*a search for ways of transplanting our current social and business mechanisms to a world in which communication is primarily telecommunication*" [15]. Now over three decades later, it is clear that the search has been fruitful.

On being asked in a 2011 interview how today's world would be different without public key cryptography, Merkle answered [69]:

> *It would be more difficult and more expensive to provide secure communications and more difficult to provide authentication information in this global network that we have. It would still be possible, but it would be more difficult and we'd have to work harder to get it.*

This raises the point that public key cryptography greatly simplifies and facilitates encryption and data authentication, compared to alternatives that use solely symmetric-key cryptography. This is important because in practice, high cost and complexity often result in solutions or services not being developed or used (especially those intended to be free, or for non-technical users). However, some systems that rely solely on symmetric key management techniques have succeeded in practice. One massive-scale example is GSM (discussed earlier); another is the Kerberos authentication infrastructure [64]. Notably, both are centralized systems with corresponding architectures and assumptions (e.g., trust in a central authority). Indeed, symmetric-key only alternatives (albeit with drawbacks) exist for various security mechanisms, and were much discussed in the late 1970s, with Merkle introducing several including digital signature schemes based solely on symmetric-key techniques [57].

While public key cryptography has enabled convenient and ubiquitous symmetric key encryption by greatly simplifying key management, it is harder to prove that in its absence, symmetric key encryption would not have enjoyed broad adoption through support by some other (less convenient, more costly) means to manage its keying material. Public key cryptography's role as an enabler nonetheless appears to have been enormous—and as Diffie has pointed out, it is not evident that its key management potential was recognized by the GCHQ in their original discovery reports. One may also observe that, given government policy at the time to suppress rather than promote use of strong cryptography, there is no reason to believe the UK government would have publicized their findings on public key cryptography, in the absence of the discoveries by Diffie, Hellman, and others.

The digital signature is an underrated star in this story. Public key signatures and the co-star they enable, public key certificates, are central to modern authentication schemes for both entities and data objects including software updates. They facilitate authenticated key management in SSH, TLS, VPNs, and secure email and secure messaging schemes. For a variety of reasons, the use of public key signatures as a legally recognized replacement for

handwritten signatures has been less of a success, and to date still largely a failure if judged against grand visions of cryptographers in the 1990s. However, the overall success of digital signatures for authentication applications has been spectacular.

Public key cryptography has resulted in major changes to computer and communication technologies. As discussed, in applications such as secure messaging, security and privacy have been greatly enhanced, but with relatively little noticeable impact on our lives. In other cases, the technology changes have, in turn, had large impacts that are social, economic, political, and even environmental in nature. For example, telecommuting is unlikely to have received broad employer support without secure remote access technology; public key has thus arguably impacted working habits, work-life relationships, and location choices for both individual homes and business offices. The March 2020 escalation of the COVID-19 pandemic due to the SARS-CoV-2 virus highlighted the critical importance of secure and dependable remote communications infrastructure for society, with this infrastructure relied on more heavily than ever before in history, and beyond its prior use by a substantially smaller fraction of work-from-home employees [10]. Nonetheless to this date, the most visible impact of public key cryptography on society has generally been viewed as its enablement of e-commerce, including online financial transactions and shopping (with payment by traditional credit cards or newer electronic methods). Whether or not new applications of greater prominence emerge, the impact of public key cryptography on society has clearly been immense.

Acknowledgements. I thank those whose input and feedback improved this paper, including: David Barrera, Xavier de Carné de Carnavalet, Roger Dingledine, Martin Hellman, Susan Landau, Alfred Menezes, Ray Ozzie, Bart Preneel and Rebecca Slayton. The author is a Canada Research Chair in Authentication and Computer Security, and acknowledges NSERC for funding that chair and a Discovery Grant.

References

[1] H. Abelson, R.J. Anderson, S.M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, M. Green, S. Landau, P.G. Neumann, R.L. Rivest, J.I. Schiller, B. Schneier, M. Specter, D.J. Weitzner. Keys under doormats: Mandating insecurity by requiring government access to all data and communications. *J. Cybersecurity* 1(1):69–79, 2015. Shorter version in: *Commun. ACM* 58(10):24–26, 2015.

[2] B. Anderson, S. Paul, D.A. McGrew. Deciphering malware's use of TLS (without decryption). *J. Computer Virology and Hacking Techniques* 14(3):195–211, 2018.

[3] B. Anderson and D.A. McGrew. TLS beyond the browser: Combining end host and network data to understand application behavior. Internet Measurement Conference (IMC), 2019.

[4] D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, H. Bos. Highly resilient peer-to-peer botnets are here: An analysis of Gameover Zeus. MALWARE 2013.

[5] A. Biryukov, I. Pustogarov, F. Thill, R.-P. Weinmann. Content and popularity analysis of Tor hidden services. ICDCS Workshops, 2014.

[6] D.L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms (Technical Note). *Commun. ACM* 24(2):84–88, Feb 1981.

[7] D. Chaum, A. Fiat, M. Naor. Untraceable electronic cash. Crypto 1988, pp.319-327.

[8] N. Christin. Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. WWW 2013.

[9] CNBC staff, March 2015. "15 years after the Nasdaq's peak: Look how it's changed". `https://www.cnbc.com/15-years-after-nasdaqs-peak-look-how-its-changed/`

[10] C. Duffy. Big tech firms ramp up remote working orders to prevent coronavirus spread. CNN Business, 12 March 2020. Online.

[11] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt and D. Stebila. A formal security analysis of the Signal messaging protocol. IEEE European Symp. on Security and Privacy (EuroS&P), 2017.

[12] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), Internet Engineering Task Force, May 2008.

[13] developerWorks Lotus. The history of Notes and Domino. November 14, 2007. `https://www.ibm.com/developerworks/lotus/library/ls-NDHistory/`

[14] S. Dechand, A. Naiakshina, A. Danilova, M. Smith, In encryption we don't trust: The effect of end-to-end encryption to the masses on user perception. IEEE European Symp. on Security and Privacy (EuroS&P), 2019.

[15] W. Diffie. The first ten years of public key cryptography. *Proceedings of the IEEE* 76(5):560–577, 1988.

[16] W. Diffie. Information Security—Before and After Public Key Cryptography. Talk, Computer History Museum, 26 Jan 2005. Video (1hr 16min): `https://www.youtube.com/watch?v=1BJuuUxCaaY`

[17] W. Diffie. 2015 Turing Award Lecture, 16 Aug 2016, San Francisco. Video (1hr 29min): `https://amturing.acm.org/vp/diffie_8371646.cfm`

[18] W. Diffie and M.E. Hellman. Multiuser cryptographic techniques. *AFIPS National Computer Conference*, 109–112, Jun 1976.

[19] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Trans. on Information Theory* 22(6):644–654, Nov 1976.

[20] W. Diffie, M.E. Hellman. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE* 67(3):397–427, 1979.

[21] W. Diffie, P.C. van Oorschot, M.J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography* 2(2):107–125, 1992.

[22] R. Dingledine, N. Mathewson, P.F. Syverson. The second-generation onion router. USENIX Security 2004.

[23] J.A. Donenfeld. WireGuard: Next generation kernel network tunnel. NDSS, 2017.

[24] S. Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, L. Repka. S/MIME Version 2 Message Specification. IETF RFC 2311 (Informational), March 1998.

[25] T.A. ElGamal. *Cryptography and Logarithms over Finite Fields*. PhD thesis, Electrical Engineering, Stanford University, June 1984.

[26] T.A. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. Proceedings of Crypto 1984, pp.10–18. Extended version: *IEEE Trans. on Information Theory* 31(4):469–472, July 1985.

[27] J.H. Ellis. The history of non-secret encryption. *Cryptologia* 23(3):267–273, 1999. Originally written 1987 (internal to UK CESG); preface by Cliff Cocks, Jan 1999.

[28] M. Gardner. A new kind of cipher that would take millions of years to break. Mathematical Games column, *Sci. American*, pp.120–124, Aug 1977.

[29] S. Ghosh, A. Das, P. Porras, V. Yegneswaran, A. Gehani. Automated categorization of onion sites for analyzing the darkweb ecosystem. KDD 2017.

[30] G. Giudici, A. Milne, D. Vinogradov. Cryptocurrencies: market analysis and perspectives. *Journal of Industrial and Business Economics* 47(1):1–18.

[31] C. Gkantsidis, T. Karagiannis, P. Rodriguez, M. Vojnovic. Planet scale software updates. ACM SIGCOMM 2006.

[32] Google. Transparency Report (HTTPS encryption on the web). `https://transparencyreport.google.com/https/overview`

[33] L.C. Guillou, M. Ugon, J.-J. Quisquater. The smart card: A standardized security device dedicated to public cryptology. Ch.12 (pp.561–613) in [74], 1992.

[34] D. Hellman and M.E. Hellman. *A New Map for Relationships: Creating True Love at Home & Peace on the Planet*. New Map Publishing, 2016.

[35] M.E. Hellman. Cybersecurity, nuclear security, Alan Turing, and illogical logic. *Commun. ACM* 60(12):52–59, 2017. Turing lecture paper.

[36] M.E. Hellman, B.W. Diffie, R.C. Merkle. Cryptographic apparatus and method. US patent 4,200,770. Filed 6 Sept 1977. Issued 29 Apr 1980.

[37] M.E. Hellman, R.C. Merkle. Public key cryptographic apparatus and method. US patent 4,218,582. Filed 6 Oct 1977. Issued 19 Aug 1980.

[38] R. Holz, J. Amann, O. Mehani, M. Wachs, M.A. Kaafar. TLS in the wild: An internet-wide analysis of TLS-based protocols for electronic communication. NDSS 2016.

[39] W. Huang, A. Ganjali, B.H. Kim, S. Oh, D. Lie The state of public Infrastructure-as-a-Service cloud security. *ACM Computing Surveys* 47(4) 68:1–68:31, 2015.

[40] International Card Manufacturers Association. Global plastic card industry grows to $27B. January 23, 2020.

[41] D. Johnson, A.J. Menezes, S.A. Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International J. of Information Security* 1(1):36–63, 2001.

[42] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, T. Kivinen. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296 (Internet Standard), Internet Engineering Task Force, Oct 2014.

[43] S.T. Kent. Internet Privacy Enhanced Mail. *Commun. ACM* 36(8):48–60, 1993.

[44] L.M. Kohnfelder. A method for certification. MIT Lab for Computer Science, 1978. This corresponds to pp.39–43 in: L.M. Kohnfelder, *Toward a Practical Public-Key Cryptosystem*, B.Sc. thesis (EE Dept), MIT, 1978.

[45] P. Kotzias, S. Matic, R. Rivera, and J. Caballero. Certified PUP: Abuse in Authenticode code signing. ACM CCS, 2015.

[46] P. Kotzias, A. Razaghpanah, J. Amann, K.G. Paterson, N. Vallina-Rodriguez, J. Caballero. Coming of age: A longitudinal study of TLS deployment. Internet Measurement Conference, 2018.

[47] D. Kravitz. Digital signature algorithm. US patent 5,231,668. Issued 27 July 1993.

[48] S. Landau. Under the radar: NSA's efforts to secure private-sector telecommunications infrastructure. *J. National Security Law & Policy* vol.7, pp.411–442, 2014.

[49] J. Lang, A. Czeskis, D. Balfanz, M. Schilder, S. Srinivas. Security Keys: Practical cryptographic second factors for the modern web. Financial Cryptography, 2016.

[50] Steven Levy. *Crypto*. Penguin Books, 2001.

[51] S. Machani, R. Philpott, S. Srinivas, J. Kemp, J. Hodges. FIDO UAF Architectural Overview. 28 Nov 2017 (v1.2 Review Draft). `https://fidoalliance.org/specs/fido-uaf-v1.2-rd-20171128/FIDO-UAF-COMPLETE-v1.2-rd-20171128.pdf`.

[52] A. Mani, T. Wilson-Brown, R. Jansen, A. Johnson, M. Sherr. Understanding Tor usage with privacy-preserving measurement. Internet Measurement Conference, Oct 2018.

[53] K.M. Martin. *Everyday Cryptography*. Oxford University Press, Second edition, 2017.

[54] D. McCoy, H. Dharmdasani, C. Kreibich, G.M. Voelker, S. Savage. Priceless: The role of payments in abuse-advertised goods. ACM CCS 2012.

[55] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[56] R.C. Merkle. Secure communications over insecure channels. *Commun. ACM* 21(4):294–299, Apr 1978.

[57] R.C. Merkle. *Secrecy, Authentication, and Public Key Systems*. PhD thesis, Electrical Engineering, Stanford University, June 1979.

[58] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Unrefereed paper, October 2008.

[59] A. Nag, J. McGeever. Foreign exchange, the world's biggest market, is shrinking. Reuters, 11 February 2016. Online.

[60] National Bureau of Standards (US). Cryptographic Algorithms for Protection of Computer Data During Transmission and Dormant Storage: Solicitation of Proposals. Federal Register (38 FR 12763), 15 May 1973.

[61] National Bureau of Standards (US). Encryption Algorithms for Computer Data Protection: Reopening of Solicitation. Federal Register (39 FR 30961), 27 August 1974.

[62] National Bureau of Standards (US). Encryption Algorithm for Computer Data Protection: Request for Comments. Federal Register (40 FR 12134–12139), 17 March 1975.

[63] A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder. *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. Princeton University Press, 2016.

[64] P.C. van Oorschot *Computer Security and the Internet: Tools and Jewels*. Springer, 2020. Online: `https://people.scs.carleton.ca/~paulv/toolsjewels.html`

[65] H. Orman. *Encrypted Email: The History and Technology of Message Privacy*. Springer Briefs in Computer Science, 2015.

[66] G. Owen, N. Savage. Empirical analysis of Tor hidden services. *IET Information Security* 10(3):113–118, 2016.

[67] Ray Ozzie. Personal communication, 12 March 2020.

[68] Jon Plutte. Whitfield Diffie Interview. 28 Mar 2011, Computer History Museum, Mountain View, CA. 17 page transcript.

[69] Jon Plutte. Ralph Merkle: 2011 Fellows Interview. 11 Mar 2011, Computer History Museum, Mountain View, CA. 10 page transcript.

[70] E. Rescorla. *SSL and TLS: Design and Building Secure Systems*. Addison-Wesley, 2001.

[71] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. IETF RFC 8446, August 2018.

[72] R.L. Rivest, A. Shamir, L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21(2):120–126, 1978. A version is cited by Gardner [28] as: MIT Lab for Computer Science, Technical Memo 82, Apr 1977.

[73] C. Rossow, C.J. Dietrich. ProVeX: Detecting botnets with encrypted command and control channels. DIMVA 2013.

[74] G.J. Simmons (editor). *Contemporary Cryptology: The Science of Information Integrity*. IEEE Press, 1992.

[75] G.J. Simmons. A survey of information authentication. Ch.7 (pp.379–419) in [74], 1992.

[76] G.J. Simmons. How to insure that data acquired to verify treaty compliance are trustworthy. Ch.13 (pp.615–630) in [74], 1992.

[77] J.C. Snader. *VPNs Illustrated: Tunnels, VPNs, and IPsec*. Addison-Wesley, 2005.

[78] J. Somorovsky, M. Heiderich, M. Jensen, J. Schwen, N. Gruschka, L.L. Iacono. All your clouds are belong to us: Security analysis of cloud management interfaces. ACM CCS Workshops (CCSW), 2011.

[79] S. Srinivas, D. Balfanz, E. Tiffany, A. Czeskis. Universal 2nd Factor (U2F) Overview. FIDO Alliance Proposed Standard. 11 April 2017. `https://fidoalliance.org/specs/fido-u2f-v1.2-ps-20170411/FIDO-U2F-COMPLETE-v1.2-ps-20170411.pdf`.

[80] Standard & Poor's. *Standard & Poor's 500 Guide: 2001 Edition*. McGraw Hill, 2001. (This edition gives stock statistics as of 7 Oct 2000.)

[81] S.T. Sun, K. Beznosov. The devil is in the (implementation) details: An empirical analysis of OAuth SSO systems. ACM CCS, 2012.

[82] K. Thomas, D.Y. Huang, D.Y. Wang, E. Bursztein, C. Grier, T. Holt, C. Kruegel, D. McCoy, S. Savage, G. Vigna. Framing dependencies introduced by underground commoditization. WEIS 2015.

[83] S. Vaudenay, M. Vuagnoux. About machine-readable travel documents. *J. Physics: Conf. Series* 77, 2007. See also ICAO Doc 9303: `https://www.icao.int/publications/Pages/doc-series.aspx`

[84] Arnd Weber. Enabling crypto: How radical innovations occur. *Commun. ACM* 45(4):103–107, Apr 2002.

[85] Arnd Weber. An interview with Ralph Merkle. 18 May 1995, edited 16 Jan 2002 to add early drafts of Merkle's "Puzzles". Transcript: `http://www.itas.kit.edu/pub/m/2002/mewe02a.htm` (ITAS, Karlsruhe, Germany).

[86] G. Welchman. *The Hut Six Story*. M&M Baldwin, 2018. 1st edition 1982, McGraw-Hill.

[87] WhatsApp. WhatsApp encryption overview (technical white paper). December 19, 2017. `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`

[88] H. Williams and W. Diffie. A.M. Turing Award Oral History Interview with Whitfield Diffie. 15 Sept 2017, Portola Valley, CA. Video (2hr 27min): `https://amturing.acm.org/interviews/diffie_8371646.cfm`. Transcript (41 pages): `https://amturing.acm.org/pdf/DiffieTuringTranscript.pdf`.

[89] H. Williams and M. Hellman. A.M. Turing Award Oral History Interview with Martin Hellman. 19 May 2017, Palo Alto, CA. Video (2hr 40min): `https://www.youtube.com/watch?v=ydYowmPaZRw&feature=youtu.be`. Transcript (47 pages): `https://amturing.acm.org/pdf/HellmanTuringTranscript.pdf`

[90] G. Wurster, P.C. van Oorschot. Self-signed executables: Restricting replacement of program binaries by malware. USENIX HotSec, 2007.

[91] T. Ylönen. SSH—secure login connections over the Internet. USENIX Security, 1996.

[92] A.L. Young, M. Yung. Cryptovirology: Extortion-based security threats and countermeasures. IEEE Symp. Security and Privacy, 1996.

[93] A.L. Young, M. Yung. Cryptovirology: The birth, neglect, and explosion of ransomware. *Commun. ACM* 60(7):24–26, 2017.

[94] P.R. Zimmermann. *The Official PGP Users Guide*. MIT Press, 1995.

[95] M. E. Zurko. IBM Lotus Notes/Domino: Embedding security in collaborative applications. Pages 607–622 in: L.F. Cranor and S. Garfinkel (editors), *Security and Usability: Designing Secure Systems That People Can Use*, O'Reilly Media, 2005.