SURVEY OF CRYPTOGRAPHIC DEVELOPMENTS

ANNIE BRYAN

1. Introduction

Cryptography, from the Greek *kryptós* (meaning "hidden" or "secret") and *graphein* (meaning "to write"), is the study of communication techniques that permit two or more parties to securely transmit information. The field of cryptography has evolved, from early substitution ciphers to the Enigma machine of World War II to today's digital communication.

A cryptographic scheme has three main goals: confidentiality, integrity, and authentication. The goal of confidentiality, first addressed in a 1976 U.S. Air Force Study [BL76], states that only the intended receiver should be able to read a message. The goal of integrity, formalized by Clark and Wilson [CW87] in their paper comparing commercial and military computer security policies, states that a message should not be able to have been altered. The goal of authentication, which has no known origin author and likely evolved over time, states that the receiver of a message should know who sent it.

Many early schemes outlined in [LP87] such as the Caesar cipher, the substitution cipher, and the Vigenère cipher, were easily decryptable, meaning that someone other than the intended receiver could decipher the messages. Thus, these ciphers lacked confidentiality. With the one-time pad cipher, messages are encrypted using a pad the length of the message, so it can only be deciphered and read by someone who knows the exact pad used in the encryption. Thus, the one-time pad introduced confidentiality, as discussed in [LP87]. However, since the process required agents to generate keys the length of the message that were only used once, it was time-consuming and computationally expensive. During World War II, the German military used the Enigma Machine to send encrypted messages that couldn't be read by the Allies. The Enigma Machine was finally cracked by Alan Turing and other researchers. Their discovery was considered by many to be the most important victory of the war. More recent work has been done, such as that by Rejewski [Rej80] and Gillogly [Gil95], in analyzing and breaking the Enigma cipher.

Another major contribution to the field came in 1976, when Whitfield Diffie and Martin Hellman [DH76] published their paper introducing *Diffie-Hellman key exchange*. One of the first public-key protocols, Diffie-Hellman key exchange is a method of securely exchanging cryptographic keys over a public or insecure channel. This was an improvement over previous encrypted communication, which required sharing keys by physical means or using a private channel. However, their scheme didn't authenticate either party, and thus it was subject to man-in-the-middle attacks. A *man-in-the-middle attack* (see [Mal19]) is a cyberattack in which a malicious third party intercepts the network between two communicating parties without their knowledge, and can read and alter messages sent over the network.

In 1978, Ronald Rivest, Adi Shamir, and Leonard Adleman [RSA78] published a public-key cryptosystem known as RSA, which I define in Definition 9. Their protocol assures confidentiality, integrity, and authentication, which I prove in Proposition 4.1.

Elliptic Curve Cryptography (ECC) was developed independently by Miller [Mil86] in 1986 and Koblitz [Kob87] in 1987. ECC improves upon RSA by having a smaller key. Thus, it is more efficient, yet maintains the same level of encryption.

In Section 2, I present background information, some basic definitions, and motivation. In Section 3, I define the Diffie-Hellman key exchange protocol and illustrate the protocol with an example. In Section 4, I present RSA public-key cryptography and give an example. Finally, in Section 5, I describe Elliptic Curve Cryptography. In Section 6, I conclude by describing some of the challenges that the field of cryptography still faces today. Specifically, I will address quantum computing, how it may render today's cryptographic methods ineffective, and what post-quantum cryptography may look like.

2. Background

The primary motivating scenario for the cryptographic schemes in this paper is a scenario with two agents, Alice and Bob[1], who wish to communicate over a channel. If this channel is insecure or public, another (adversarial) agent Eve may be listening to messages sent on the channel, and could potentially intercept them. To protect messages from being read by anyone other than the intended recipient, I introduce the notion of *symmetric key cryptography*.

[1]The first mention of agents named Alice and Bob was in the 1978 paper by Rivest, Shamir, and Adleman [RSA78], and became a common trope of the field within a few years.

Definition 1. *Symmetric key cryptography*, often called private-key cryptography, consists of an encryption scheme $E : K \times M \to C$ and a decryption scheme $D : K \times C \to M$, where M is a set of messages, C is a set of ciphertexts, and K is a secret key that only the two communicating parties know.

The security of any symmetric-key cryptosystem relies on the (lack of) invertibility of E_K. Namely, given E_K, it must be hard to compute D_K, without knowledge of K.

The "hardness" of a problem refers to its *computational hardness* or *computational complexity*, which is a measure of the amount of computing resources required by an algorithm to solve a problem. The three most important classes of complexity classes for motivating cryptographic schemes are *P*, *NP*, and *NP-Hard*.

Definition 2. *P* (Polynomial time) is a complexity class that contains all decision problems that can be solved in polynomial time with respect to the size of the input.

Definition 3. *NP* (Non-deterministic Polynomial time) is a complexity class that contains all decision problems that can be verified in polynomial time.

Definition 4. *NP-Hard* is a complexity class that contains all decision problems that are at least as hard as all problems in NP. In other words, a problem H is NP-Hard if every problem in NP can be reduced to H in polynomial time.

Such functions that are easy to calculate but hard to invert are known as *one-way functions*.

Definition 5. [Matb] A function f is a *one-way function* if:

(1) The description of f is publicly known.
(2) Given x, it is easy to compute $f(x)$.
(3) Given y in the range of f, there is no efficient algorithm to find an x such that $f(x) = y$.

Example 1. Factorization problem: $f(p, q) = pq$ for primes p, q.

Example 2. Discrete logarithm problem: $f(p, g, x) = g^x \pmod{p}$.

If one can prove that computing D_K is sufficiently hard without knowledge of K, then one has proved that the cryptosystem achieves confidentiality. Conversely, if one can find an efficient algorithm for computing D_K or K given E_K, then this is sufficient to show that the cryptosystem lacks confidentiality and has been "broken".

One such encryption function that is easily invertible is the Caesar cipher, which has an encryption function of

$$E_n(x) = (x + n) \pmod{26}$$

and a decryption function of

$$D_n(x) = (x - n) \pmod{26}.$$

Even without knowledge of the key n, one could decipher $c := E_n(x)$ by Algorithm 1, which has a time complexity of $O(1)$. Since there are a limited number of shifts, they can each be tested, one at a time. This type of trial-and-error approach, in which an attacker systematically iterates over all possible keys, is known as a *brute-force attack*.

Algorithm 1: Efficient decryption of the Caesar cipher

```
for 1 ≤ n' ≤ 25 do
    m' ← (c − n') (mod 26);
    if m' is a readable message then
        return (n', m');
    end
end
```

Another example of a brute-force attack is guessing a 4-digit code (e.g. a bank account PIN or a phone password). Since there are $10^4 = 10,000$ possible values that the password could be, it would take the attacker at most $10,000$ tries before guessing correctly. This is why phones have a maximum number of tries before it will lock you out, or require you to wait a few minutes before trying again.

3. Diffie-Hellman Key Exchange System

Suppose Alice and Bob want to communicate via some symmetric-key cryptographic protocol. In order for Alice and Bob to agree on a shared secret key K for encryption and decryption, they can either share this key using physical means (as was the case until the 1970's), or they can use the *Diffie-Hellman Key Exchange System* to agree on a key. Let p be a prime number and let g be an element of the finite field $GF(p)$.

Definition 6. [Mata] For a prime p, the *finite field* order p, denoted $GF(p)$, (often written $\mathbb{F}_p$) is the field of residue classes modulo p, with p elements $\{0, 1, \ldots, p-1\}$. Finite fields by definition satisfy the *field axioms*:

1.	Associativity of addition	$(a+b)+c = a+(b+c)$
2.	Commutativity of addition	$a+b = b+a$
3.	Additive identity	$a+0 = a$
4.	Additive inverses	$a+(-a) = 0$
5.	Associativity of multiplication	$(ab)c = a(bc)$
6.	Commutativity of multiplication	$ab = ba$
7.	Multiplicative identity	$a(1) = a$
8.	Multiplicative inverse	$a(a^{-1}) = 1$ (for all $a \neq 0$)
9.	Distributivity	$a(b+c) = ab+ac$

Example 3. For $p = 3$, the finite field order 3, $GF(3)$ consists of the 3 elements $\{0, 1, 2\}$, which satisfy the additive and multiplicative tables given in Table 1 and Table 2.

$+$	0	1	2
0	$0 + 0 \pmod 3 = \mathbf{0}$	$0 + 1 \pmod 3 = \mathbf{1}$	$0 + 2 \pmod 3 = \mathbf{2}$
1	$1 + 0 \pmod 3 = \mathbf{1}$	$1 + 1 \pmod 3 = \mathbf{2}$	$1 + 2 \pmod 3 = \mathbf{0}$
2	$2 + 0 \pmod 3 = \mathbf{2}$	$2 + 1 \pmod 3 = \mathbf{0}$	$2 + 2 \pmod 3 = \mathbf{1}$

TABLE 1. Additive table of the finite field $GF(3)$

$\times$	0	1	2
0	$0 \times 0 \pmod 3 = \mathbf{0}$	$0 \times 1 \pmod 3 = \mathbf{0}$	$0 \times 2 \pmod 3 = \mathbf{0}$
1	$1 \times 0 \pmod 3 = \mathbf{0}$	$1 \times 1 \pmod 3 = \mathbf{1}$	$1 \times 2 \pmod 3 = \mathbf{2}$
2	$2 \times 0 \pmod 3 = \mathbf{0}$	$2 \times 1 \pmod 3 = \mathbf{2}$	$2 \times 2 \pmod 3 = \mathbf{1}$

TABLE 2. Multiplicative table of the finite field $GF(3)$

Definition 7. The *Diffie-Hellman Key Exchange System*, a protocol for securely exchanging a key over a public or insecure channel, is defined as the following set of steps:

(1) Alice and Bob publicly agree on values of p and g.
(2) Alice picks a number a uniformly at random from $\{1, \ldots, p-1\}$.
(3) Alice computes $A = g^a \pmod{p}$, which she sends to Bob.
(4) Bob picks a number b uniformly at random from $\{1, \ldots, p-1\}$.
(5) Bob computes $B = g^b \pmod{p}$, which he sends to Alice.
(6) Alice computes $K_A = B^a \pmod{p}$.
(7) Bob computes $K_B = A^b \pmod{p}$.

Proposition 3.1. The Diffie-Hellman Key Exchange System yields the same key for Alice and Bob ($K_A = K_B$).

Proof. Substituting the value Bob calculated in step (5) into the key Alice calculated in step (6) yields

$$\begin{aligned} K_A &= B^a \pmod{p} \\ &= \left(g^b \pmod{p}\right)^a \pmod{p} \\ &= g^{ab} \pmod{p}. \end{aligned} \tag{1}$$

Substituting the value Alice calculated in step (3) into the key Bob calculated in step (7) yields

$$\begin{aligned} K_B &= A^b \pmod{p} \\ &= \left(g^a \pmod{p}\right)^b \pmod{p} \\ &= g^{ab} \pmod{p}. \end{aligned} \tag{2}$$

Since the values of (1) and (2) are the same, this concludes our proof. □

Example 4. Let $p = 83$ and $g = 7$.
Suppose Alice chooses $a = 15$. She calculates $A = 7^{15} \pmod{83} = 31$.
Suppose Bob chooses $b = 40$. He calculates $B = 7^{40} \pmod{83} = 12$.
Given Bob's value of $B = 12$, Alice calculates $s = 12^{15} \pmod{83} = 75$.
Given Alice's value of $A = 31$, Bob calculates $s = 31^{40} \pmod{83} = 75$.

Proposition 3.2. The key $K = K_A = K_B$ generated by the protocol defined in Definition 7 cannot be easily calculated by anyone without knowledge of a or b.

Proof. Consider an agent Eve who knows all publicly agreed-upon values (p, g) and all values sent over the network (A, B), but does not know secret values generated by Alice and Bob (a, b). Currently, there is no known way to compute K from A and B without first

computing a or b. Thus, Eve must solve the discrete logarithm problem (Ex. 2), which is a trapdoor function, in order to compute $K = B^{\log_g A} \pmod p = A^{\log_g B} \pmod p$.

If the prime p is slightly less than 2^c for some integer c, then exponentiation (calculating $g^a \pmod p$ or $g^b \pmod p$) requires at most $2c$ multiplications, while taking a log (calculating $\log_g A \pmod p$ or $\log_g B \pmod p$) requires around $p^{1/2} = 2^{c/2}$ operations. Thus, the complexity of computing logarithms grows exponentially. This completes the proof. □

Although the Diffie-Hellman Key Exchange protocol allows Alice and Bob to establish a secret shared key over a public channel, one drawback of the protocol is that it is subject to *man-in-the-middle (MITM) attacks*. A MITM attack is a situation in which an adversarial agent Eve is able to insert herself between Alice and Bob by hijacking their connection and establishing her own independent connections with each of them. Once this has been done, Eve can relay messages between Alice and Bob, making it appear to them that they are communicating with each other. Since a symmetric-key protocol enabled by the Diffie-Hellman Key Exchange Scheme does not authenticate the agent sending the message, in the next section I will introduce public-key cryptography and cryptographic signatures.

4. RSA Public Key Cryptography

Suppose Alice wants to sign a message m such that Bob can be assured that it came from Alice. This introduces the idea of a *cryptographic signature*, which cannot be forged, and thus guarantees that the message came from a known sender. This introduced the notion of a *public-key* cryptosystem.

Definition 8. *Public key cryptography*, often called asymmetric-key cryptography, consists of an encryption scheme $E : K_e \times M \to C$ and a decryption scheme $D : K_d \times C \to M$. The encryption key K_e is public knowledge, while the decryption key K_d is known only to the individual who owns the key. Each individual who wishes to receive messages has their own private decryption key K_d and a corresponding public encryption key K_e.

To implement signatures in a public-key cryptosystem such as RSA, signatures must use one-way functions, also called *trapdoor functions*, which are easy to compute in one direction, but nearly impossible to invert. Let E_A, D_A, E_B, and D_B be the encryption and decryption schemes for Alice and Bob, respectively. Recall that E_A and D_A are

inverses, while E_B and D_B are inverses. Additionally, recall that E_A and E_B are public information, while D_A and D_B are known only to Alice and Bob, respectively.

Definition 9. I the RSA Public-Key Cryptography scheme [RSA78], a protocol for signature generation and verification, as the following set of steps:

(1) Alice computes $s = D_A(m)$.
(2) Alice computes $c = E_B(s)$.
(3) Alice sends c to Bob over an insecure channel.
(4) Bob computes $s = D_B(c)$.
(5) Bob computes $m = E_A(s)$.

Proposition 4.1. The protocol defined in Definition 9 ensures confidentiality, integrity, and authentication.

Proof. I start with the value c sent over the public channel, where $c = E_B(s) = E_B(D_A(m))$ for some message m.

Bob is assured that the message came from Alice, since D_A is private to Alice, so she is the only one who can decrypt using D_A. Therefore, this protocol ensures authentication.

Bob is assured that the message has not been altered in any way since Alice sent it, since to do so would require knowledge of D_A. Therefore, this protocol ensures integrity.

Bob obtains Alice's original message, since $E_A(D_B(c)) = E_A(s) = m$, so this protocol ensures correctness.

The function D_B is private to Bob, so he is the only one who can read Alice's message. Therefore, this protocol ensures confidentiality. This completes the proof. □

The functions used in [RSA78] take advantage of the difficulty of the *factoring problem*: given the product of two large primes $n = p \cdot q$, find p and q. Clearly, calculating n from p and q is easy. However, without knowledge of p or q, deducing them given only n is much harder. The specific encryption and decryption functions are defined as

$$C \equiv E(M) \equiv M^e \pmod{n}, \tag{3}$$

$$M \equiv D(C) \equiv C^d \pmod{n}, \tag{4}$$

where d is an integer which is relatively prime to $\Phi(n) = (p-1)(q-1)$ and e is the multiplicative inverse of d, modulo $(p-1)(q-1)$.

The fastest method of factoring integers under 100 decimal digits is the *Quadratic sieve algorithm*, developed by Pomerance [Pom82] in

1982. To factor an integer n, the Quadratic sieve algorithm runs in $e^{(1+o(1))\sqrt{\ln n \ln\ln n}} = L_n\left[\frac{1}{2}, 1\right]$.

Another method of factoring is *Fermat's factoring algorithm*, developed by Pierre de Fermat and described by Lehman [Leh74] in 1974. When the factors are of roughly similar size, Fermat's factoring algorithm is efficient.

5. Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) was developed independently by Miller [Mil86] in 1986 and Koblitz [Kob87] in 1987. Its one-way function is scalar multiplication of points about an elliptic curve.

Definition 10. An *elliptic curve* E_F defined over a field F is the set of solutions $(x, y) \in F^2$ to the equation

$$y^2 = x^3 + ax + b \tag{5}$$

for some constant values $a, b \in F$, such that the cubic on the right side of the equation has no multiple roots.

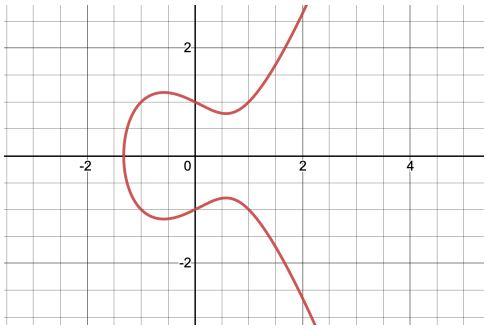

Figure 1. An elliptic curve $y^2 = x^3 - x + 1$

An example of such an elliptic curve with $a = -1, b = 1$ is denoted in Figure 1. Note that an elliptic curve is symmetric about the x-axis. Specifically, if (x_1, y_1) is a solution to Equation (5), then $(x_1, -y_1)$ is a solution as well.

Consider the following operations performed on points P_1, P_2 satisfying Equation (5). An example is illustrated in Figure 2. Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ with finite values x_1, y_1, x_2, y_2 such that $y_1 \neq -y_2$. Draw a line through P_1 and P_2 (if $P_1 = P_2$, draw the line tangent to E_F at P_1). Let $P_3 = (x_3, y_3)$ be the negative of the third point of intersection $(x_3, -y_3)$ between the line and E_F.

To compute P_3, we need to find the line through P_1 and P_2, which is

$$y - y_1 = m(x - x_1),$$

where

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

is the slope of the line if $P_1 \neq P_2$, and

$$m = \frac{\delta E_F}{\delta x}\bigg|_{(x_1,y_1)} = \frac{3x_1^2 + a}{2y_1}$$

is the slope of the tangent line at P_1, which we use if $P_1 = P_2$.

To find the point of intersection between the curve E_K and the line between P_1 and P_2, we set them equal and solve for x and y, which gives us:

$$x_3 = -x_1 - x_2 + m^2, \qquad y_3 = -y_1 + m(x_1 - x_3), \tag{6}$$

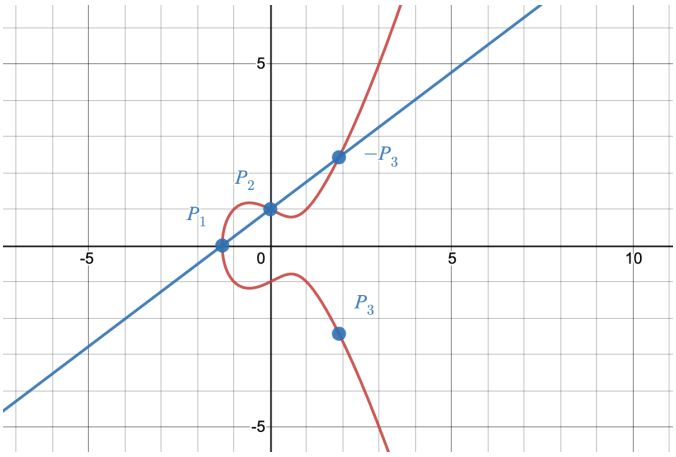

FIGURE 2. An elliptic curve $y^2 = x^3 - x + 1$ with $P_1 = (-1.32471, 0)$, $P_2 = (0, 1)$, $P_3 = (1.89456, -2.43017)$

Using formulas (6), I will define $f : F^2 \times F^2 \to F^2$ as a function that applies the above operation. Furthermore, I will define *scalar multiplication* as the repeated application of f to a point. In other words, to calculate kP for some integer k:

(1) Calculate $2P = Q_1 = f(P, P)$
(2) Calculate $3P = Q_2 = f(P, Q_1)$
(3) Calculate $4P = Q_3 = f(P, Q_2)$
$\vdots$
$(k-1)$ Calculate $kP = Q_{k-1} = f(P, Q_{k-2})$

It is easy to compute kP given k and P, via repeated squaring. However, to compute k given P and kP, the naïve or brute-force solution

would be to compute multiples of P until arriving at kP. In practice, the value of k is typically 256 bits. Thus, a brute-force solution requires repeating the above procedure at most 2^{256} times, which is infeasible as it is extremely time-intensive. The basis of the security of ECC is this trapdoor function known as the *Elliptic Curve Discrete Logarithm Problem*.

Definition 11. Let E be an elliptic curve over the finite field $GF(q)$, where $q = p^n$ for prime p. The *Elliptic Curve Discrete Logarithm Problem (ECDLP)* states: Given points $P, Q \in E$, find an integer k such that $Q = kP$, if such k exists.

One algorithm that solves the ECDLP is *Pollard's rho algorithm*, developed by Pollard [Pol78] in 1978, which computes the index of any integer relative to a given root of a prime p in $O(\sqrt{p})$ operations.

An improved algorithm is the *Pohlig-Hellman algorithm*, proposed by Pohlig and Hellman [PH78] in 1978. In the general case, the Pohlig-Hellman runs in $O(\sqrt{p})$, but improves in the case where $p-1$ has only small prime factors, in which it runs in $O(\log_2(p))$ complexity.

ECC is one of the most commonly used encryption standards on the Internet today, especially in transactions of cryptocurrency such as Bitcoin and Ethereum.

6. Challenges and Future Work

Quantum computing is a rapidly-emerging technology that relies on quantum mechanics for computation. Regular computers encode data in binary digits, or bits, that can take on a value of 0 or 1. By contrast, quantum computers use *qubits*, where a single qubit is able to encode more than two states [Stu18]. Therefore, algorithms written for quantum computers can run significantly faster than those written for regular computers.

Today's cryptographic techniques rely on one-way functions that cannot be inverted. To iterate through all possible values, also known as a brute-force solution, would take unreasonably long. However, quantum computing would greatly speed this process up. Therefore, today's cryptographic techniques would be rendered essentially useless by quantum computers.

While some researchers have made progress at building small-scale quantum computers, many challenges still exist. It may take several more years before quantum computers are able to scale to a size capable of destroying our cryptographic schemes.

References

[Leh74] R. Sherman Lehman. "Factoring large integers". In: *Mathematics of Computation* 28.126 (1974), pp. 637–646. DOI: 10.1090/s0025-5718-1974-0340163-2.

[BL76] D. Elliott Bell and Leonard J. La Padula. "Secure Computer System: Unified Exposition and multics interpretation". In: (1976). DOI: 10.21236/ada023588. URL: https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/early-cs-papers/bell76.pdf.

[DH76] W. Diffie and M. Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[PH78] S. Pohlig and M. Hellman. "An improved algorithm for computing logarithms over¡tex¿GF(p)¡/tex¿and its cryptographic significance (Corresp.)" In: *IEEE Transactions on Information Theory* 24.1 (1978), pp. 106–110. DOI: 10.1109/TIT.1978.1055817.

[Pol78] J. M. Pollard. "Monte Carlo Methods for Index Computation (mod P)". In: *Mathematics of Computation* 32.143 (July 1978), pp. 918–924. DOI: 10.2307/2006496.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (Feb. 1978), pp. 120–126. DOI: 10.1145/359340.359342.

[Rej80] M. Rejewski. "An application of the theory of permutations in breaking the Enigma cipher". In: *Applicationes Mathematicae* 16.4 (1980), pp. 543–559. DOI: 10.4064/am-16-4-543-559.

[Pom82] Carl Pomerance. "Analysis and comparison of some integer factoring algorithms". In: *Computational Methods in Number Theory, Part I* (1982), pp. 89–139.

[Mil86] V.S. Miller. "Use of elliptic curves in cryptography". In: *Advances in Cryptology - CRYPTO '85 Proceedings*. Ed. by H.C. Williams. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 417–426. DOI: 10.1007/3-540-39799-X_31.

[CW87] David D. Clark and David R. Wilson. "A comparison of commercial and military computer security policies". In: *1987 IEEE Symposium on Security and Privacy* (1987). DOI: 10.1109/sp.1987.10001.

[Kob87] N. Koblitz. "Elliptic curve cryptosystems". In: *Mathematics of Computation* 48.177 (Jan. 1987), pp. 203–209. DOI: 10.1090/S0025-5718-1987-0866109-5.

[LP87] Dennis Luciano and Gordon Prichett. "Cryptology: From caesar ciphers to public-key cryptosystems". In: *The College Mathematics Journal* 18.1 (Jan. 1987), pp. 2–17. DOI: 10.1080/07468342.1987.11973000.

[Gil95] James J. Gillogly. "CIPHERTEXT-ONLY CRYPTANALYSIS OF ENIGMA". In: *Cryptologia* 19.4 (1995), pp. 405–413. DOI: 10.1080/0161-119591884060. URL: https://doi.org/10.1080/0161-119591884060.

[Stu18] R. Stubbs. *Quantum Computing and its Impact on Cryptography*. Apr. 2018. URL: https://www.cryptomathic.com/news-events/blog/quantum-computing-and-its-impact-on-cryptography.

[Mal19] Avijit Mallik. "Man-in-the-middle-attack: Understanding in simple words". In: *Cyberspace: Jurnal Pendidikan Teknologi Informasi* 2.2 (2019), p. 109. DOI: 10.22373/cj.v2i2.3453.

[Mata] Wolfram MathWorld. *Finite field*. URL: https://mathworld.wolfram.com/FiniteField.html.

[Matb] Wolfram MathWorld. *Trapdoor one-way function*. URL: https://mathworld.wolfram.com/TrapdoorOne-WayFunction.html.