*cryptography*

*Article*

# Cryptanalysis and Improvement of ECC Based Authentication and Key Exchanging Protocols †

**Swapnoneel Roy * and Chanchal Khatwani**

School of Computing, University of North Florida, Jacksonville, FL 32224, USA; n01029842@ospreys.unf.edu
* Correspondence: s.roy@unf.edu; Tel.: +1-904-620-2182

† This paper is an extended version of our paper published in CICN 2015: Khatwani, C.; Roy, S. Security Analysis of ECC Based Authentication Protocols. In the Proceedings of the 2015 International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, India, 12–14 December 2015; pp. 1167–1172.

Academic Editor: Kwangjo Kim
Received: 3 April 2017; Accepted: 29 May 2017; Published: 13 June 2017

**Abstract:** Elliptic curve cryptography (ECC) is extensively used in various multifactor authentication protocols. In this work, various recent ECC-based authentication and key exchange protocols are subjected to threat modeling and static analysis to detect vulnerabilities and to enhance them to be more secure against threats. This work demonstrates how currently-used ECC-based protocols are vulnerable to attacks. If protocols are vulnerable, damage could include critical data loss and elevated privacy concerns. The protocols considered in this work differ in their usage of security factors (e.g., passwords, pins and biometrics), encryption and timestamps. The threat model considers various kinds of attacks including denial of service, man in the middle, weak authentication and SQL injection. Countermeasures to reduce or prevent such attacks are suggested. Beyond cryptanalysis of current schemes and the proposal of new schemes, the proposed adversary model and criteria set forth provide a benchmark for the systematic evaluation of future two-factor authentication proposals.

**Keywords:** elliptic curve cryptography; authentication protocols; key exchanging protocols

## 1. Introduction

In a cyber environment, user authentication enables a perimeter device (firewall, proxy server, VPN server, remote access server, etc.) to decide whether to approve a specific user's request to gain entry to the network. The authentication is generally two-way, meaning both parties (provider and user) authenticate themselves to each other [1]. Client authentication needs security for remote login, while the client's program is trying to communicate with the server's program over insecure networks, like the Internet. The identity and a secret password of a client are used for mutual authentication and access control. However, a password can be compromised during transmission if an efficient scheme is not followed.

Elliptic curve cryptography (ECC) is a widely-used technique in multi-factor authentication [1]. ECC is a public key encryption technique based on elliptic curve theory that can be used to create smaller keys, which yields faster and more efficient algorithms as a result. It generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the product of very large prime numbers. The technology can be used in conjunction with most public key encryption methods, such as RSA (RSA is made of the initial letters of the last names of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1978.) and Diffie–Hellman [2]. ECC was developed to reduce computational costs while providing the same level of security as other similar operations (e.g., modular exponentiation). ECC finds applications in authentication protocols involving smart cards, RFIDs, wireless networks, digital signatures and other authentication techniques [3–14]. However, the computational cost of one bilinear pairing (an important operation of ECC) is about twice as high as that of one modular exponentiation operation at the same security level [15]. Therefore, the computationally-intensive nature of ECC leaves a security loophole in the protocols that use it. An attacker can force the server or client to repeatedly perform ECC operations in order to clog them, resulting in one or all of them wasting resources by performing unnecessary computations. We are looking at two different types of schemes that use ECC, one being multi-factor authentication and the other being key-exchange.

Multi-factor authentication is an approach to authentication in which the user is required to provide more than one form of verification in order to prove his/her identity and allow access to the system. It takes advantage of a combination of several forms of authentication. Three major forms include verification by: (1) something a user knows (such as a password); (2) something the user has (such as a smart card or a security token); and (3) something the user is (such as a biometric characteristic). Due to their increased complexity, authentication systems that use multi-factor verification are harder to break than those that use a single factor [1,16–18].

Key exchange (also known as key establishment) is a cryptographic method by which secret keys are exchanged between two parties, allowing use of a cryptographic algorithm. Public key cryptography, or asymmetric cryptography, is a cryptographic system that uses pairs of public and private keys. Each party has its own public key and private key. The message is encrypted using the public key and decrypted using one's private key. The public key is distributed, and the private key is known only to the owner. If the sender and receiver wish to communicate with each other, then a secret key is shared between them in order for communication to take place. Symmetric key algorithms use the same cryptographic keys for both encryption and decryption of text. The key exchange problem is how to exchange information so that no third party can obtain a copy. Typically, this has required trusted couriers or a secure channel.

Radio frequency identification (RFID) is a means of automatic identification that uses radio waves to detect, track, identify and thus manage a variety of objects. Although RFID technology has been around for more than half a century, only recently have RFID security and privacy issues begun to attract attention from both academic and corporate research communities. RFID (radio frequency identification) technology has become ubiquitous because of its low cost. Since it is used in many fields, any vulnerability detected in RFID technology raises a threat in data privacy. Similarly, the potential for smart cards is enormous. However, by far the most serious problem for smart cards is the attacks that exploit security vulnerabilities caused by poor design or implementation. These vulnerabilities tend to be easy to exploit and replicate and are, therefore, shared among the hackers community.

### *1.1. Problem Statement, Goal and Contribution*

ECC is a multi-factor encryption technique currently used by the U.S. government, Tor, Bitcoin, iMessage and SSL/TLS. Such multi-factor authentication is needed to provide high level security, but the introduction of more factors introduces more vulnerability in the protocols. Our goal is to detect these vulnerabilities before they are exploited. The root causes of many common vulnerabilities like CPU resource-exhaustion, stack overflow, etc., are often design flaws rather than programming (implementation) errors [19]. Therefore, developers perform static analysis on protocols to identify design flaws in order to ensure security before a program (software product) is launched.

Several multi-factor authentication protocols that involve RFIDs, smart cards, wireless networks or digital signatures rely on ECC operations for their security. However, the computational cost of one bilinear pairing (an important operation of ECC) is about twice as high as that of one modular exponentiation at the same security level. Therefore, the computationally-intensive nature of ECC leaves a security loophole in the protocols. Hence, some level of protection should be added to ECC-based protocols to guarantee near-total security against various kinds of attacks, such as denial of service, man in the middle and database attacks.

As observed in this research, the vulnerabilities of ECC-based protocols are not recognized frequently. The first goal of this work is to perform static analysis on the underlying vulnerabilities and security threats that exist in ECC-based protocols that are implemented in RFIDs and smart cards. The second goal is to design possible countermeasures to defeat the identified vulnerabilities in these protocols. The contribution of this research is to provide the basis for future work in improving the security of ECC-based protocols using dynamic analysis. The results of this work will contribute to the cybersecurity community in a considerable way.

### 1.2. Organization of This Paper

The rest of the paper is organized as follows. Section 2 provides background materials on ECC. Section 3 describes the cryptanalysis algorithms and discusses the type of protocols that were investigated in this work. Section 4 describes in detail the work done in this paper to cryptanalyze various protocols. It describes the attacks that were carried out on the protocols and proposes countermeasures for each attack. Finally, Sections 5 and 6 discuss the results, conclusions and suggest future directions, respectively.

## 2. ECC Background

ECC is one of the most accomplished and widely used, however least understood, cryptography tools [20]. It is the future generation of public key cryptography. It provides significantly more security than first-generation public key cryptography systems like RSA [20]. ECC is a technique in public key cryptography set on the algebraic arrangement of elliptic curves over finite fields. Compared to non-ECC cryptography, ECC provides equivalent security with smaller keys [21,22]. The elliptic curve cryptosystem [21] was initially proposed as a basis for public key cryptosystems, and it has proven to be an important unit of current cryptography [23]. ECC utilizes the mathematics of elliptic curves. The security of ECC lies in the complexity of working the elliptic curve discrete logarithm problem. An analysis of ECC theory and its computational problems are stated below.

As shown in Figure 1, elliptic curves $(E_q(a,b))$ are a set of points defined by the solutions to the equation $y^2 \equiv x^3 + ax + b \pmod{q}$, where $a$ and $b$ are elements of the field $k$ together with a point at infinity $\mathbf{O}$ [24]. There is also a condition such that $4a^3 + 27b^3 \neq 0 \pmod{q}$ where $q$ is a prime number [24]. This equation must be satisfied for the elliptic curve to have a well-defined group structure. This forms an additive cyclic group $E = \{(x,y) \in E_q(a,b)\} \cup \{\mathbf{O}\}$, where $\mathbf{O}$ serves as an additive identity element of the group [24]. If $P$ is a point in $E$ and k is a positive integer, then the point multiplication is computed by repeated addition, such as $k \cdot P = P + P \cdots + P$, where $k$ is a large integer and P is added to itself $k$ times.

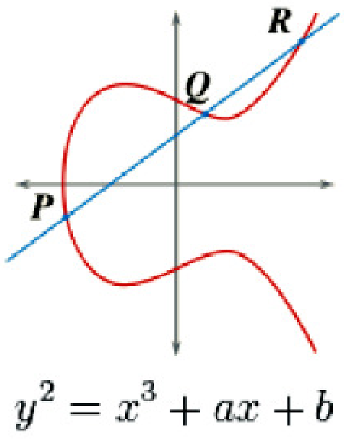

**Figure 1.** Elliptic curve addition (figure redrawn based on [14]).

### 2.1. Computational Nature of ECC

ECC is a computationally-intensive operation. Its scalar multiplication is one-way, making it computationally infeasible to trace the original number. For example: let $P$ be a point in $E$, and let $Q$ be a point such that $Q = kP$. The elliptic curve discrete log problem is the following: knowing the values of $P$ and $Q$, determine the value of $k$. If the modulus $q$ is large, the ECDLP (For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible: this is the "elliptic curve discrete logarithm problem" (ECDLP).) is computationally infeasible. ECC is based on this problem. Even if $P$ and $Q$ are known, determining $k$ such that $Q = kP$ ($kP$ and $k \cdot P$ have the same meaning in ECC multiplication) is computationally infeasible. Hence, the elliptic curve discrete log problem makes $k$ difficult to compute.

## 3. Framework for Cryptanalysis

In this section, general algorithms and conditions for the attacks that were successfully performed on the protocols chosen for this work are described.

### 3.1. Clogging Attack

The mechanism for most password authentication protocols is that the client (usually a memory stick, RFID or smart card reader) sends its credentials to the server, which then performs certain mathematical operations to verify those credentials. The protocols usually work in multiple phases. The phases in which client and server authentication take place will be discussed in Section 4 after each protocol has been individually considered.

As shown in Figure 2, the main idea of the clogging attack is the interception of the message that contains the login credentials between the client and the server [25]. This message is unencrypted in some protocols and encrypted in others. It might or might not contain a timestamp. The attacker replays the intercepted message several times to force the server to perform computationally-intensive operations (in the case of [25], modular exponentiation), thus forcing the server to waste its time and resources. Legitimate users are denied services in that way. Algorithm 1 depicts this type of attack.

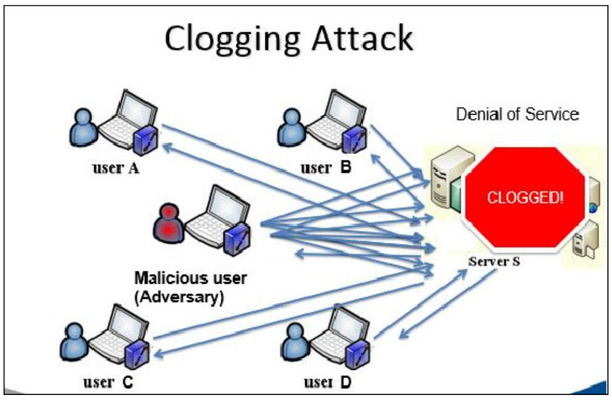

**Figure 2.** The clogging attack.

### 3.2. Application of Algorithm 1

The clogging conditions we analyze in this work are based on the computational and resource intensiveness of the operations in elliptic curve cryptography (ECC). The ECC operations performed by the majority of authentication based on ECC are:

1. Bilinear pairing
2. Scalar multiplication in group $G$
3. Map-to-point conversion

**Algorithm 1:** The general algorithm for the clogging attack [25].

```
Intercept login message from client to server
if Timestamp is present then
    Modify timestamp to match requirements
else
    Keep message as is
end if
while The server is not completely clogged! do
    Replay the message to the server
end while
```

Let $T_p$, $T_s$ and $T_{map}$ respectively be the time taken to perform a single bilinear pairing, scalar multiplication and map-to-point conversation, respectively. It has been shown in [26] that:

1. $T_p > T_s > T_{map}$
2. $T_p \approx 3 \times T_s$
3. $T_p \approx 4 \times T_{map}$

Further, let $T_{modex}$ be the time taken by one modulo exponentiation operation. It has been shown [15] that $T_p \approx 2 \times T_{modex}$ for the same level of security. The operation modular exponentiation has been shown to be very computationally intensive [25]. In fact, $T_{modex}$ has been shown to be approximately a hundred times that of normal addition, multiplication and bitwise XOR operations (see Figure 3). We can, therefore, conclude that all of the ECC-based operations' bilinear pairing, scalar multiplication and map-to-point conversation are quite computationally intensive.

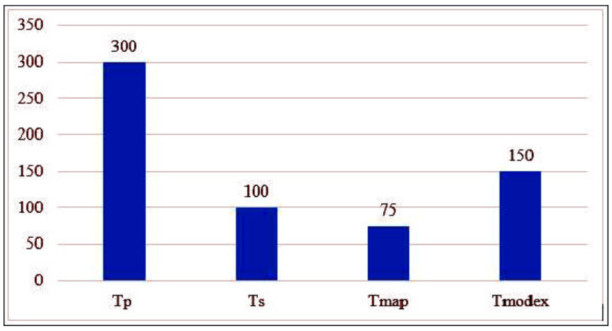

**Figure 3.** Comparison of ECC and Modex operations.

Therefore, a protocol that uses the ECC operation has a vulnerability to the clogging attack, a form of DoS in which the attacker exploits the computational intensiveness nature of ECC operations.

### *3.3. Database Attack*

According to many experts, databases are still not secured properly in most organizations [27]. Database attacks go unnoticed as it takes less than 10 seconds to hack in and out of a database. Therefore, it is no surprise that many database attacks go unnoticed by organizations until long after the data have been compromised. Attackers use simple methods to break into databases, such as exploiting weak authentication, using default passwords and capitalizing on known vulnerabilities [27].

This analysis focuses on database connections that are weak and hence open to vulnerabilities. The front end client-server authentication stores passwords in the server's back-end databases. If any password is compromised, then the database schema becomes vulnerable to attack, which makes the protocol insecure (as explained in Algorithm 2). Passwords (or their hashed versions) are often stored in relational databases. The most common way to get illegitimate access to a database is by making a copy of the database (or individual rows) by a technique called SQL injection. SQL injection attacks occur where the fields available for user input let SQL statements through to query the database directly. Outside of the client, web applications typically are the weakest link [27]. URL parameters or POST parameters of a web page are manipulated to contain malicious SQL statements, which are interpreted by the SQL database [28].

**Algorithm 2:** The general algorithm for the database attack.

```
Intercept data access layer from application to back-end
if Encryption is present then
    Break the encryption to gain access to the database
else
    Access the database
end if
while The data is not corrupted and stolen do
    Inject malicious statements
end while
```

Internal attacks should also never be underestimated. There have been several insider attacks that came as a result of a malicious user possessing more system privileges than the user should have had [27]. Databases are often accessible from inside organizations, and passwords can readily be found in the source code or server configuration files. This makes it easy for employees to access data and save it to a local disk or even transfer it to an external medium.

In the following chapters, database attacks (illustrated in Figure 4) to which recently-used protocols are vulnerable are considered.

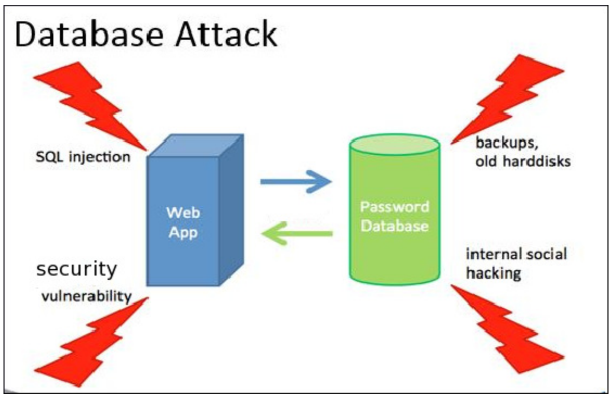

**Figure 4.** The database attack.

## *3.4. Man-In-The-Middle Attack*

As shown in Figure 5, a man-in-the-middle attack can be used against many cryptographic protocols. A man-in-the-middle attack requires an attacker to have the ability to both monitor and alter or inject messages into a communication channel. One example is active eavesdropping, in which the attacker makes independent connections with the victims and relays messages between them to make them believe they are talking directly to each other over a private connection, when in fact the entire conversation is controlled by the attacker. The attacker can intercept all messages passing between the two victims and inject new ones. A few ECC-based protocols claim to be secure against man-in-the-middle attacks. However, ECC protocols are still vulnerable to man-in-the-middle attacks, as shown in the next section.

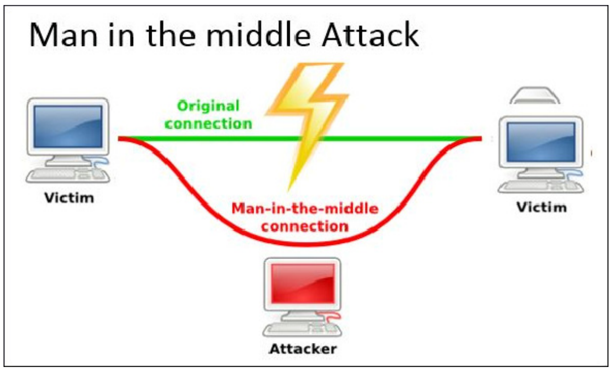

**Figure 5.** The man-in-the-middle (MITM) attack.

### *3.5. Application of Algorithm 3*

A variant of the Diffie–Hellman algorithm that uses elliptic curve cryptography, elliptic curve Diffie–Hellman (ECDH), is an anonymous key agreement protocol that allows two parties, each having an elliptic curve public-private key pair, to establish a shared secret over an insecure channel (for a detailed description and illustration of ECDH, please see [29–32]). This shared secret may be directly used as a key, or it may be used to derive another key, which can then be used to encrypt subsequent communications using a symmetric key cipher. The following example will illustrate how a key is established. Suppose Alice wants to establish a shared key with Bob, but the only channel available for them may be eavesdropped by a third party. Initially, the domain parameters for ECDH (that is, $(p, a, b, G, n, h)$ in the prime case or $(m, f(x), a, b, G, n, h)$ in the binary case) must be agreed upon. Furthermore, each party must have a key pair suitable for elliptic curve cryptography, consisting of a private key $d$ (a randomly-selected integer in the interval $[1, n-1]$) and a public key $Q$ (where $Q = dG$, that is the result of adding $G$ together $d$ times where $G$ is the generator point). Let Alice's key pair be $(d_A, Q_A)$ and Bob's key pair be $(d_B, Q_B)$. Each party must know the other party's public key prior to execution of the protocol.

**Algorithm 3:** The general algorithm for the man-in-the-middle attack.

```
Intercept communication between two parties
if TTP is present then
    Gains access and possibly alters the communication between two parties who believe they
    are directly communicating with each other
else
    Acts as an intruder who relays and alters the communication between two parties
end if
while The communication is not ended do
    Relay
end while
```

Alice computes $(x_k, y_k) = d_A Q_B$. Bob computes $(x_k, y_k) = d_B Q_A$. The shared secret is $x_k$ (the $x$ coordinate of the computed point). Most standardized protocols based on elliptic curve Diffie–Hellman (ECDH) derived a symmetric key from $x_k$ using some hash-based key derivation function. The shared secret calculated by both parties is equal, because $d_A Q_B = d_A d_B G = d_B d_A G = d_B Q_A$ (by the property of ECC). The only information about her private key that Alice initially exposes is her public key. Therefore, no party other than Alice can determine Alice's private key, unless that party can solve the elliptic curve discrete logarithm problem. Bob's private key is similarly secure. No party other than Alice or Bob can compute the shared secret, unless that party can solve the elliptic curve Diffie–Hellman problem [29].

The public keys are either static (and trusted, say via a certificate) or ephemeral (ECDHE (Elliptic curve Diffie-Hellman Exchange (ECDHE) is an anonymous key agreement protocol that allows two parties, each having an elliptic curve public-private key pair, to establish a shared secret over an insecure channel.)). Ephemeral keys are temporary and not necessarily authenticated, so if authentication is desired, authenticity assurances must be obtained by other means. Authentication is necessary to avoid man-in-the-middle attacks. If one of Alice's or Bob's public keys is static, then man-in-the-middle attacks are thwarted. Static public keys provide neither forward secrecy, nor key-compromise impersonation resilience, among other advanced security properties. Holders of static private keys should validate the other public key and should apply a secure key derivation function to the raw Diffie–Hellman shared secret to avoid leaking information about the static private key [29].

## 4. Cryptanalysis of the Protocols

This section describes each of the protocols selected for analysis in this work, the attacks on them and ways of preventing the attacks. The following generic notations have been used to describe the protocols in this section. Specific notations for each individual protocol have been illustrated while describing each of them:

- $\mathbf{Z}_q^*$ denotes the finite field over $q$
- $\otimes$ denote (bitwise) exclusive OR
- $A \rightarrow B$: $M$ denotes the propagation of the message $M$ from user $A$ to user $B$
- $||$ denotes the concatenation operation
- In cryptography, a nonce is an arbitrary number that may only be used once

### *4.1. Choice of Protocols*

The protocols chosen for analysis fall into the broad category of multi-factor authentication protocols. All of them employ user ID and password for authentication. Selection of these protocols is based on differences in the second authentication factor (smart cards, RFIDs, memory drives, etc.) and the tools used to provide confidentiality (encryption, nonce, timestamp, etc.), as shown in Table 1.

**Table 1.** Summary of the differences in the protocols.

| Protocol | Type | Factor Used | Confidentiality |
|---|---|---|---|
| Moosavi | Authentication | RFID | Usage of Random Numbers |
| Xu | Authentication | Smart Card | Usage of Random Numbers |
| He | Authentication | RFID | Usage of Random Numbers |
| Hui | Authentication | Password | Usage of Encryption |
| Ammayappan | Key Exchange | – | Trusted Third Party |

### *4.2. Moosavi et al.'s Protocol for RFID Implant Systems*

The first protocol considered is that of [33]. It is a mutual authentication scheme for an RFID implant system. The authors in [33] claim that their protocol is immune to various attacks including denial of service (DoS). However, their protocol is inherently vulnerable to clogging attacks (a form of DoS) that apply the algorithm of [25]. Most of the precursor protocols to that of [33] are vulnerable to clogging attacks. In this section, the mathematical basis that makes the protocols vulnerable to clogging attack is identified, and a possible fix is suggested.

### 4.2.1. Review of the Protocol

Moosavi et al.'s protocol works in three phases: reader authentication and verification, tag identification and tag verification. Figure 6 shows the notations used for the protocol.

This protocol enables the two communicating parties, an RFID implant tag and a reader, to respectively verify and ensure each other's identity. The assumption is that the communication between the tag and the reader is not secure [1]. The protocol is shown in Algorithm 4.

**Algorithm 4:** Moosavi et al.'s protocol for RFID implant systems.

```
READER AUTHENTICATION AND VERIFICATION

Reader R
1. Step A1. Choose a random number $r_1 \in \mathbf{Z}_n$ and computes $R_1 = r_1 \cdot P$ as its public key.
2. Step A2. Initialize $i_1$ to 1.
3. Step A3. $R \rightarrow T$: $\{R_1, i_1\}$.
4. Step A4. Increment $i_1$ by $r_1$.

Tag T
1. Step A5. Verify if $i_1 \geq i_2$ ($i_2$ is initialized to 0).
2. Step A6. If the above is true, then set $i_2$ to $i_1$.
3. Step A7. Compute $r_3 = X(r_2 \cdot P) * Y(R_1)$, where $*$ is a non-algebraic operation over the abscissa
   of $(r_2 \cdot P)$ and the ordinate of $R_1$.
4. Step A8. $T \rightarrow R$: $\{r_3\}$.

Reader R
1. Step A9. Compute $R_2 = r_1 \cdot ID_t + r_3 \cdot s_3$.
2. Step A10. $R \rightarrow T$: $\{R_2\}$.

Tag T
1. Step A11. Verify if $(R_2 - r_1 \cdot ID_t) r_3^{-1} \cdot P = ID_r$.
2. Step A12. Reader $R$ gets verified if the above is true.

TAG IDENTIFICATION

Reader R
1. Step I1. Choose $r_s \in \mathbf{Z}_n$, a random integer.
2. Step I2. $R \rightarrow T$: $\{r_s\}$.

Tag T
1. Step I3. Verify if $r_s \neq 0$. If yes, then calculate $s_2 = f(X(s_1)) \cdot P$.
2. Step I4. Choose a random integer $k$ and compute curve point $(x, y) = k \cdot G$
3. Step I5. Compute $d = x \mod n$
4. Step I6. Verify if $d = 0$. If yes, recompute $d$ using a different $k$.
5. Step I7. Compute value of ID as $ID_t = (Mb(X(s_1)) * Mb(X(s_2))) \cdot P$.
6. Step I8. Compute $c = k \cdot (Hash(ID_t) + X(s_1) * d) \mod n$.
7. Step I9. Verify if $c = 0$. If yes, recompute $c$ using a different $k$.
8. Step I10. $T \rightarrow R$: $\{ID_t, (d, c)\}$.

TAG VERIFICATION

Reader R
1. Step V1. Choose a random integer $r_s \in \mathbf{Z}_n$.
2. Step V2. Compute public key $p_r = r_s \cdot P$.
3. Step V3. Verify if $d, c \in \mathbf{Z_n}$.
4. Step V4. If true, compute $h = Hash(ID_t)$.
5. Step V5. Compute $w = c^{-1} \mod n$, $u_1 = zw \mod n$, and $u_2 = dw \mod n$ respectively.
6. Step V6. Compute curve point $(x, y) = u_1 \cdot P + p_r$.
7. Step V7. Verify if $r = x \mod n$. If true, authenticate tag $T$.
```

### 4.2.2. Analysis of Moosavi et al.'s Protocol

This protocol is for an RFID implant system that has applications in microchip implants. The identities are the tag that is implanted and the reader that verifies (and authenticates) the tag. Communication between the tag and the reader is through an insecure network. Additionally, the reader is connected to a database through a secured channel, so the reader and database are considered to be a single entity for analysis purpose [1]. The protocol uses ECC techniques twice, once during tag identification (Step I4 of Algorithm 4) and once during tag verification (Step V6 of Algorithm 4). However, in the course of this work, it became clear that the tag verification phase has a dangling step in Step V6. The use of variable $u_2$ in the verification process is not mentioned in this protocol. Furthermore, from [33], the description of verification variable $r$ of Step V7 is not mentioned in this protocol.

- $G$: a group of order $q$ on an elliptic curve having the order $n$
- $P$: a primitive element or the base point of $G$
- $s_1, s_2$: each tag keeps two secret points $s_1, s_2 \in E(F_g)$, which will change over time. These secret points will be varied each time the tag is successfully identified
- $ID_t$: the tag's identification number or ID
- $s_3$: each reader keeps a secret point $s_3 \in \mathbf{Z}_n$, which will change over time. The secret point will be varied each time the reader is successfully authenticated
- $ID_r = s_3 \cdot P$: the reader's public key
- $r_s, i_1, i_2$: random numbers in $\mathbf{Z}_n$
- $h$: a lightweight hash function
- $(d, c)$: a signature generated by the tag during its identification phase

**Figure 6.** Notations for Moosavi et al.'s protocol.

### 4.2.3. Clogging Attack on Moosavi et al.'s Protocol

The security analysis of [33] has been performed as in Section 5 of their paper [33]. The adversary $\mathcal{A}$ has the same power as assumed by Moosavi et al. $\mathcal{A}$ needs to be able to read and modify the contents of messages over an insecure channel during the tag identification and tag verification phases of the protocol [1]. The line of attack in this work is a denial of service where the objective of $\mathcal{A}$ is to bring down the whole RFID system. There are two steps (a map-to-point conversion in Step V6 and a scalar multiplication in Step V2) where ECC techniques are used that $\mathcal{A}$ can target [1]. However, since it would be more profitable to render the reader $R$ useless, the line of attack chosen here will target bringing down the reader $R$, such that it would not be able to provide service to legitimate tags.

1. $\mathcal{A}$ intercepts a valid message of $T \rightarrow R$: $\{ID_t, (d, c)\}$ from Step I10.
2. Since the message is unencrypted, $\mathcal{A}$ can always change the $(d, c)$, such that $d, c \in \mathbf{Z_n}$ holds (though $\mathcal{A}$ might not need to).
3. $\mathcal{A}$ just relays $\{ID_t, (d, c)\}$ to $R$.

The following is performed by the reader $R$:

1. Step V1. Choose a random integer $r_s \in \mathbf{Z}_n$.
2. Step V2. Compute public key $p_r = r_s \cdot P$.
3. Step V3. Verify if $d, c \in \mathbf{Z_n}$.
4. Step V4. If true, compute $h = Hash(ID_t)$.
5. Step V5. Compute $w = c^{-1} \mod n$, $u_1 = zw \mod n$, and $u_2 = dw \mod n$ respectively.
6. Step V6. Compute curve point $(x, y) = u_1 \cdot P + p_r$.
7. Step V7. Verify if $r = x \mod n$. If true, authenticate tag $T$.

$\mathcal{A}$ would make the reader $R$ repeat Steps V1 through V7 to compute the ECC steps several times. $\mathcal{A}$ can change all of the incoming login request messages from any legitimate tag to $R$. Since the ECC steps are computationally intensive [15], the victimized reader $R$ spends considerable computing resources doing useless ECC computations (a map-to-point conversion in Step V6 and a scalar multiplication in Step V2) along with the other steps, V1, V3 through V5 and V7, rather than any real work. Thus, $\mathcal{A}$ clogs $R$ with useless work and therefore denies any legitimate tag (user) any service. $\mathcal{A}$ needs only an ID of a single valid tag to perform the clogging attack [1].

It should be noted that even DoS-resilient mechanisms (e.g., timeout or locking the user account for a period of time after a predefined number of login failures) are introduced on reader $R$'s side; it may be not a real obstacle for attacker $\mathcal{A}$, as it can initialize new sessions with different intercepted identities in an interleaving manner. Hence, $\mathcal{A}$ can potentially perform the above attack procedure continuously. If distributed DoS attacks are launched based on this strategy, the consequences will be more serious [1].

#### 4.2.4. Proposed Countermeasures from the Attack

At the beginning of the authentication phase, the reader could check whether the network address of the tag is valid. It has to know the network addresses of all of the registered legitimate tags. In spite of that, $\mathcal{A}$ could spoof the network address of a legitimate tag and replay the tag verification message. To prevent this spoofing, a cookie exchange step could be added at the beginning of the tag verification phase of Moosavi et al.'s protocol [1]. This step has been designed as in the well-known Oakley key exchange protocol [34].

1. The tag $T$ chooses a pseudo-random number $n_1$ and sends it along with the message $\{ID_t, (d, c)\}$.
2. The reader $R$, upon receiving the message, acknowledges the message and sends its own cookie $n_2$ to $T$.
3. The next message from $T$ must contain $n_2$, else $T$ rejects the message and the tag verification request.

Security analysis of the fix: Had $\mathcal{A}$ spoofed $T$'s IP address, $\mathcal{A}$ would not get $n_2$ back from $R$. Hence, $\mathcal{A}$ succeeds only in having $R$ send back an acknowledgment, but not in launching the computationally-intensive ECC-based operations. Hence, the clogging attack is avoided by these additional steps. It must be noted, however, that this process does not prevent the clogging attack, but only repels it to some extent. This fix can fully work only if $n_1$ and $n_2$ are encrypted respectively by $T$'s and $R$'s private keys for a secure communication [1].

### *4.3. Xu et al.'s Smart Card-Based Protocol*

The next protocol we look at in this work is that of Xu et al. [26]. It is an ECC-based remote authentication protocol involving smart cards. Xu et al.'s protocol in [26] is an improvement over its predecessor, Li et al.'s protocol [35], which [26] proved to be vulnerable against the off-line password guessing, user impersonation and the denial of service (DoS) attacks. The authors in [26] came up with a new protocol, which they claim is immune against all three attacks.

We briefly present Xu et al.'s protocol. We then present a clogging attack on the protocol. Most of the protocols they cite in their paper [26] are vulnerable to clogging attack. We then suggest a possible fix against clogging attack for this protocol [1].

### 4.3.1. Review of the Protocol

Xu et. al's protocol works in five phases: registration, authentication, password change, card revocation and user eviction. Figure 7 shows the notations used for the protocol.

We present the registration, authentication phases of the protocol in Algorithm 5. The other phases are omitted since they are not needed to demonstrate the clogging attack [1].

- $q$: a large prime
- $E(F_q)$: an elliptic curve over a finite field $F_q$
- $P$: the generator of a subgroup of $F_q$ with a prime order $n$
- $O$: the point on the elliptic curve at infinite, where $n \times P = O$
- $G$: a cyclic addition group generated by $P$ on the elliptic curve
- $X_s$: the secret key kept confidentially in $S$
- $P_{pub}$: $S$'s public key $(= X_s \times P)$
- $K_x$: the $x$ coordinate of the point $K$, $K = (K_x, K_y)$
- $e(P, Q)$: the bilinear pairing computation using two points $P$ and $Q$, which are in $G$
- $U_i$: the $i^{th}$ user
- $ID_i$: the identity of $U_i$
- $PW_i$: the password of $U_i$
- $V_{i1}, V_{i2}$: the password verifiers of $U_i$
- $r_i, r_i^{new}$ , $r_u$: the random numbers generated by $U_i$
- $r_s$: the random number generated by $S$
- $sk_u, sk_s$: the session keys generated by the user and the server, respectively
- $h(\cdot), h_i(,)$ $(i = 0, 1, 2)$: the one-way hash functions
- $l$: the length of one-way hash function and the security parameter
- $E_k(\cdot)/D_k(\cdot)$: the symmetric encryption/decryption function with key $k$
- $+/-$: elliptic curve point addition/subtraction
- $\Rightarrow$: a secure channel
- $\rightarrow$: an insecure channel
- $a\|b$: the concentration of two strings $a$ and $b$
- $a \oplus b$: the X-OR computation using $a$ and $b$
- $A$: a malicious attacker

**Figure 7.** Notations for Xu et al.'s protocol.

### 4.3.2. Clogging Attack on Xu et al.'s Protocol

The adversary $\mathcal{A}$ has the same power as assumed by Xu et al's [26] while exposing the weaknesses of Li et al.'s protocol [35]. $\mathcal{A}$ needs to be able to read and modify the contents of messages over an insecure channel during the authentication phase of the protocol [1].

1. $\mathcal{A}$ intercepts a valid login request ($\{CID_i, B_1, R_1\}$) from step Step A7.
2. $\mathcal{A}$ simply replays this login message to $S$.

**Algorithm 5:** Xu et al.'s authentication protocol.

```
                         REGISTRATION PHASE
                              User $U_i$
1. Step R1. Choose identity $ID_i$, password $PW_i$.
2. Step R2. Choose $r_i$
3. Step R3. Compute $HPW_i = h_0(PW_i \parallel r_i)$
4. Step R4. $U_i \rightarrow S$: $\{ID_i, HPW_i\}$ through a secure channel.
                              Server $S$
1. Step R5. Check $ID_i$, choose $N$
2. Step R6. Compute $k_i = h_0(X_s \parallel ID_i \parallel N)$ and $W_i = k_i \oplus HPW_i$.
3. Step R7. Store $\{W_i, P, P_{pub}\}$ into smart card.
4. Step R8. Store $\{ID_i, N\}$ in the server's database.
5. Step R9. $S \rightarrow U_i$: The smart card.
                              User $U_i$
1. Step R10. Upon receiving the smart card from $S$, $U_i$ enters stores $r_i$ in it.
                           AUTHENTICATION
                              User $U_i$
1. Step A1. $U_i$ inserts his smart card and inputs $ID_i$, and $PW_i$.
2. Step A2. Smart card chooses random $r_u \in \mathbf{Z}_n^*$.
3. Step A3. Compute $HPW_i = h_0(PW_i \parallel r_i)$
4. Step A4. Compute $R_1 = r_u \times P = (R_{1x}, R_{1y})$, $R_2 = r_u \times P_{pub} = (R_{2x}, R_{2y})$.
5. Step A5. Compute $B_1 = h_0((W_i \oplus HPW_i) \parallel h_0(R_{1x} \parallel R_{2x} \parallel R_{1y} \parallel R_{2y}))$.
6. Step A6. Compute $CID_i = ID_i \oplus h_0(R_{2x} \parallel R_{2y})$.
7. Step A7. $U_i \rightarrow S$: $\{CID_i, B_1, R_1\}$.
                              Server $S$
1. Step A8. Compute $R'_2 = X_s \times R_1$, $ID'_i = CID_i \oplus h_0(R'_{2x} \parallel R'_{2y})$, $k_i = h_0(X_s \parallel ID'_i \parallel N)$.
2. Step A9. Verify if $B_1 = h_0(k_i \parallel h_0(R_{1x} \parallel R'_{2x} \parallel R_{1y} \parallel R'_{2y}))$. Abort if not true, continue if true.
3. Step A10. Choose random $r_s \in \mathbf{Z}_n^*$.
4. Step A11. Compute $R_3 = r_s \times P = (R_{3x}, R_{3y})$, $K_s = r_s \times R_1 = (K_{sx}, K_{sy})$.
5. Step A12. Compute $B_2 = h_0(k_i \parallel h_0(R'_{2x} \parallel R'_{2y}))$.
6. Step A13. Compute $B_3 = h_1(R_{1x} \parallel R_{1y} \parallel R_{3x} \parallel R_{3y} \parallel ID_i \parallel S \parallel B_2 \parallel K_{sx} \parallel K_{sy})$.
7. Step A14. Compute session key $sk_s = h_2(R_{1x} \parallel R_{1y} \parallel R_{3x} \parallel R_{3y} \parallel S \parallel ID_i \parallel B_2 \parallel K_{sx} \parallel K_{sy})$.
8. Step A15. $S \rightarrow U_i$: $\{R_3, B_3\}$.
                              User $U_i$
1. Step A16. Compute $K_u = r_u \times R_3 = (K_{ux}, K_{uy})$ and $B'_2 = h_0((W_i \oplus HPW_i) \parallel h_0(R_{2x} \parallel R_{2y}))$.
2. Step A17. Verify if $B'_3 = h_1(R_{1x} \parallel R_{1y} \parallel R_{3x} \parallel R_{3y} \parallel ID_i \parallel S \parallel B_2 \parallel K_{ux} \parallel K_{uy})$. Abort if not true,
   continue if true.
3. Step A18. Compute session key $sk_s = h_2(R_{1x} \parallel R_{1y} \parallel R_{3x} \parallel R_{3y} \parallel S \parallel ID_i \parallel B_2 \parallel K_{ux} \parallel K_{uy})$.
```

**Lemma 1.** *The server $S$ is forced to perform Steps A8 through A15 if $\{CID_i, B_1, R_1\}$ is valid.*

**Proof.** The only step that could prevent the server $S$ from executing further steps is Step A9. We claim $B_1 = h_0(k_i \parallel h_0(R_{1x} \parallel R'_{2x} \parallel R_{1y} \parallel R'_{2y}))$ would always be true if $B_1$ and $R_1$ are valid. To prove our claim, we note $B_1 = h_0((W_i \oplus HPW_i) \parallel h_0(R_{1x} \parallel R_{2x} \parallel R_{1y} \parallel R_{2y}))$. Plugging in the value of $W_i = k_i \oplus HPW_i$, we have $B_1 = h_0(k_i \parallel h_0(R_{1x} \parallel R_{2x} \parallel R_{1y} \parallel R_{2y}))$. Now, $R_1 = r_u \times P = (R_{1x}, R_{1y})$, $R_2 = r_u \times P_{pub} = (R_{2x}, R_{2y})$ and $R'_2 = X_s \times R_1$. Therefore, if we have a valid $B_1$ and $R_1$, the values of $R_2$ and $R'_2$ always match, which proves the claim. □

The adversary $\mathcal{A}$ replays the message $\{CID_i, B_1, R_1\}$ several times and makes the server $S$ compute Steps A8 through A15 (by Lemma 1). These steps contain several scalar multiplications and map-to-point conversations. $\mathcal{A}$ could potentially store all of the incoming login request messages from any legitimate user to $S$ for future replay. Since the mathematical operations in the steps are computationally intensive, the victimized server spends considerable computing resources doing useless computations rather than any real work. Thus, $\mathcal{A}$ clogs $S$ with useless work and therefore denies any legitimate user any service. $\mathcal{A}$ needs only one message from a single valid user to perform the clogging attack repeatedly [1].

However, as [26] noted, the adversary $\mathcal{A}$ cannot get a legitimate access through this replay attack. Since $r_u$ will be different each time, the check $B_2' = h_0((W_i \oplus HPW_i) \parallel h_0(R_{2x} \parallel R_{2y}))$ on the user's side will not pass. However the intention of $\mathcal{A}$ here is not to gain illegitimate access, but to bring down the server by launching a clogging attack [1].

Clogging attack performed on other similar schemes: The clogging attack performed on [26] can also be performed on Li's protocol [35]. The clogging attack on Li's protocol is more effective because Li's protocol [35] uses the most computationally-intensive bilinear pairing operation. Both the protocols by [26,35] are vulnerable because the user's smart card does not encrypt the message it sends to the server for login and authentication. Thus, the adversary has the chance to manipulate the message.

Proposed countermeasures from clogging attack: Replay attacks on most smart card-based protocols are possible because their security relies on computationally-intensive operations (in this case ECC), and the messages are not encrypted by default. This vulnerability is often overlooked, since the natural result of a replay is not a DoS. An approach to reducing these attacks on Xu and Wu's protocol (and smart card based protocols in general) would be:

1. $U_i$ uses a time stamp $T$ in Step A7, and $S$ verifies it in Step A8. The time stamp also must be encrypted in some form, so that $\mathcal{A}$ cannot tamper with it.
2. $S$ checks whether multiple login requests frequently comes from the same user. This reduces the chances of a reply.

The chances of a replay are reduced, but not eliminated, because $\mathcal{A}$ can obtain many valid user IDs (they are public) and send fake login requests periodically from different IDs. Alternatively, $\mathcal{A}$ can store various (valid) login requests over time and replay them periodically [1].

Yet another way to prevent clogging attack: The mathematical basis of the protocols' vulnerability to clogging attacks is modular exponentiation. An approach to completely avoiding clogging attacks would be to encrypt all of the messages between $U_i$ and $S$. Doing so would require a key exchange step, where each user has a private key and a public key [1]. The server knows the public key and can decrypt a message encrypted by a user's private key. Thus, the server makes sure that the message is from a valid user before it computes the costly modular exponentiation. This approach comes with a cost and depends on the level of security desired. This countermeasure works for all protocols (whether or not they are smart card based) [1].

### 4.3.3. Weak Authentication and SQL Injection Attack

The proposed scheme in this paper does not mention how the ID and N are stored in the account table in the registration phase. The paper does not mention if an encryption is present at the database level. Under the assumption that the account table is accessible to an untrusted source, ID and N can be dynamically constructed in the query thus leading to a SQL injection attack. Furthermore, under the assumption that ID and N are unprotected, a weak authentication attack is possible, as well. These attacks lead to loss in confidentiality, authentication, authorization and integrity. Hence, we conclude that the database layer authentication is not strong enough to protect unknown users from gaining access to the database.

Proposed security countermeasures for the weak authentication attack: SQL injection attacks can be defeated by parsing and validating SQL communications to make sure they are not corrupted. If the protocol shows how the ID and N are stored and protected in a table, then the attack will be minimized. The weak authentication attack can be protected by adding more access layers and by enforcing strict user privileges.

### 4.3.4. Unauthorized Access Attack

In [26], in Sections 4.4, card revocation, and 4.5, user eviction, talk about a smart card being stolen and verified after being stolen. This can lead to a potential risk. If the smart card is stolen by the attacker, the user ID and the password will be accessible to the attacker. The attacker can enter malicious input and potentially hack the database server. Under the assumption that the smart card re-registration phase is unprotected, the unauthorized access attack is possible on this protocol. For example, an unauthorized client can steal the smart card or eavesdrop on the information exchanged between a legitimate client and directory server. This critical information such as user ID and password can be used to cause other major attacks.

Proposed security countermeasures against unauthorized access attack: To counter unauthorized access, card revocation should be handled by more efficient algorithms. Authentication methods, password policies and access control mechanisms provided by the directory server offer efficient ways of preventing unauthorized access. A network firewall, consisting of a hardware device, software program or a combination of the two, protects an internal computer network against malicious access from the outside. Network firewalls may also be configured to restrict access to the outside from internal users.

## *4.4. He et al.'s RFID-Based Authentication Protocol*

The next protocol we analyze in this work is due to [36]. This is an RFID-based authentication protocol to preserve identity privacy. He et al.'s protocol is an improvement over [37], which proved to be vulnerable against impersonation and insider attacks. Figure 8 shows the notations used for the protocol.

- $n, q$: two large prime numbers.
- $F(q)$: a finite field, where $q$ represents the size of the finite field.
- $(a, b)$: two parameters of an elliptic curve $E$, which is defined by the equation $y^2 = x^3 + ax + b$ over the finite field $F(q)$.
- $P$: a generator point with order $n$ of the elliptic curve $E$.
- $x_S$: the private key of the server.
- $P_S$: the public key of the server, where $P_S = x_S P$.
- $X_T$: the ID-verifier of the tag.

**Figure 8.** Notations for He et al.'s protocol.

This protocol has been proven to be immune to various attacks [36]. We present a clogging attack on the protocol and observe that it is inherently vulnerable to clogging, weak authentication and SQL injection attacks. The proposed ECC-based RFID authentication scheme consists of two phases, i.e., the setup phase and the authentication. We present the protocol in Algorithm 6.

### 4.4.1. Clogging Attack on He et al.'s Protocol

The adversary $\mathcal{A}$ has the same power as assumed by [36] while exposing the weaknesses of Liao's protocol [37]. $\mathcal{A}$ needs only to be able to read and modify the contents of messages over an insecure channel during the authentication phase. The steps taken by A to perform the clogging attack are as follows:

- $\mathcal{A}$ intercepts a valid login request $m_2 = R_2, Auth_T$ from Step A4.
- Since the message is unencrypted, $\mathcal{A}$ changes $Auth_T$ to any random garbage value $Auth_{\mathcal{A}}$.
- $\mathcal{A}$ then sends $\{R_2, Auth_{\mathcal{A}}\}$ to the server $S$.

The following is performed by the server $S$:

1. The server $S$ calculates $TK_{S1} = X_S R_2$, $TK_{S2} = r_1 R_2$ and $X'_T = (Auth\mathcal{A}_{\oplus} TK_{S2}) - TK_{S1}$.
2. The server searches its database for $X'_T$. It is not found, and therefore, the server terminates the session.

$\mathcal{A}$ would now repeat the steps several times and make the server $S$ compute the elliptic curve operations several times in Steps A5 and A6. $\mathcal{A}$ can potentially change all of the incoming login request messages from any legitimate tag to $S$. Thus, $\mathcal{A}$ clogs $S$ with useless work (ECC operations) and therefore denies any legitimate user any service. $\mathcal{A}$ only requires the value of $X_T$ of a single valid user to perform the clogging attack repeatedly.

**Algorithm 6:** He et al.'s protocol.

SET UP PHASE

1. **Step S1.** The server $S$ chooses a random number $x_S \in Z_n^*$ such that $P_S = x_S P$.
2. **Step S2.** The server $S$ chooses a random point $X_T$ on the elliptic curve $E$ for each tag.
3. **Step S3.** $S$ stores the ID-verifier $X_T$ and parameters into the tag's memory. The server also keeps $x_S$ as its private key, and stores $X_T$ into its database.

AUTHENTICATION PHASE

`Server` $S$

1. **Step A1.** Choose a random number $r_1 \in Z_n^*$ then computes $R_1 = r_1 P$.
2. **Step A2.** $S \rightarrow T$: $m_1 = R_1$.

`Tag` $T$

1. **Step A3.** Choose a new random number $r_2 \in Z_n^*$ then calculates $R_2 = r_2 P$, $TK_{T1} = r_2 P_S$, $TK_{T2} = r_2 R_1$, and $Auth_T = (X_T + TK_{T1}) \oplus TK_{T2}$.
2. **Step A4.** $T \rightarrow S$: $m_2 = R_2, Auth_T$.

`Server` $S$

1. **Step A5.** The server $S$ calculates $TK_{S1} = X_S R_2$, $TK_{S2} = r_1 R_2$, and $X_T = (Auth_T \oplus TK_{S2}) - TK_{S1}$.
2. **Step A6.**The server search its database for $X_T$. If it is not found, the server terminates the session. Otherwise, the server calculates $Auth_S = (X_T + 2TK_{S1}) \oplus (2TK_{S2})$.
3. **Step A7.** $S \rightarrow T$: $m3 = \{Auth_S\}$.

`Tag` $T$

1. **Step A8.** The tag checks whether $(X_T + 2TK_{T1}) \oplus (2TK_{T2})$ and $Auth_S$ are equal. If they are not equal, the tag terminates the session, otherwise, the server is authenticated.

Proposed countermeasures for the clogging attack: One reason for the vulnerability of this protocol is that the messages $m_1$, $m_2$, $m_3$ are not encrypted; if the adversary gains access to the message $m_2$, then repeated transmission of this message can clog the server. Furthermore, the timestamp checking is not used between the tag and the server. Although, random numbers are generated in every step and used for real-time calculation, when messages are not encrypted and timestamps are not used, the messages are open to the adversary to modify and replay. The solution could be to add a timestamp $T_i$ to $R_i$ or to $m_2$. The server can check the validity of the timestamp before it performs the elliptic curve operations.

#### 4.4.2. Weak Authentication Attack

The authors in the paper [36] do not mention how $X_T$ (ID identifier) is stored in the database. Weak authentication schemes allow attackers to assume the identity of legitimate database users. Specific attack strategies include brute force attacks, social engineering, and so on [27]. Under the assumption that $X_T$ is unprotected, the adversary can gain access to $X_T$ resulting in unauthorized data access, corruption or availability. Hence, weak authentication attack is possible assuming the database layer authentication is not strong enough to protect unknown users from gaining access to the database.

Proposed security countermeasures for the weak authentication attack: To prevent weak authentication attack, the implementation of passwords or two-factor authentication is a must [27]. Had the protocol [36] included security and protection for $X_T$, the weak authentication attack could have been avoided. The weak authentication attack can also be prevented by adding more access layers and by enforcing stricter user privileges.

#### 4.4.3. Desynchronization Attack

A desynchronization attack is a typical RFID related threat in which a tag's key stored in the back-end database and the tag's memory would not be the same, because an attacker blocks the communication between the parties. The paper [36] mentions that $X_T$ is not performed ($X_T$ is an ID identifier). Their paper claims that the proposed scheme provides scalability, availability and DoS resistance by not updating $X_T$. To provide privacy protection, most RFID authentication schemes update the tag's secret information, in the back-end database, as well as in the tag, after a successful protocol run. Therefore, synchronization of secret information between the database and the tag is crucial for subsequent authentications. The most serious threat to which an RFID tag is vulnerable is the desynchronization attack. During the past few years, RFID technology has become ubiquitous as the cost got lower; it is used in every field, and hence, this vulnerability raises a threat in the area of data protection.

Proposed security countermeasures for the desynchronization attack: One of the countermeasures is to update the $X_T$ after each run. It is an intractable task to design the lightweight RFID authentication protocol, because the security engineer must cope with the trade-offs among security, cost and performance. In the future, the security engineers can be made aware of the trade offs so as to build an efficient protocol.

### *4.5. Hui et al.'s Protocol*

The next protocol we analyze is due to Hui et al. [38]. In that paper, after pointing out the weakness of the password change phase of Islam et al. [39] and after the evaluation of several other password authentication schemes, [38] have presented a new password-based authentication and update scheme using ECC and showed that it can resist various attacks. However, in this work, the protocol is shown to be inherently vulnerable to clogging, weak authentication and SQL injection attacks.

#### 4.5.1. Review of the Protocol

Hui et al.'s protocol works in four stages: registration phase, password authentication phase, session key distribution phase and password change phase. We present the protocol in Algorithm 7. Figure 9 shows the notations used for the protocol.

**Algorithm 7:** Hui et al.'s protocol.

REGISTRATION PHASE

`Client C`

1. **Step C1.** Client $C$ chooses identity $ID_C$, $pw_C$.
2. **Step C2.** $U_C = pw_C \cdot P$.
3. **Step C3.** Client $C$ sends $ID_C$ and $U_C$ to the server $S$.

`Server S`

1. **Step C4.** Server $S$ stores $ID_C$ and $U_C$ in a write protected file.

PASSWORD AUTHENTICATION PHASE

`Client C`

1. **Step C5.** Enter $ID_C$, $pw_C$.
2. **Step C6.** Choose random number $r_C \in Z_n^*$.
3. **Step C7.** Compute $W_C = r_C \cdot pW_C \cdot U_S$, $R_C = r_C \cdot U_C = (k_x, k_y)$, $Y_C = r_C \cdot P$.
4. **Step C8.** Compute $M_1 = E_{kx}(ID_C, Y_C)$.
5. **Step C9.** Client $C$ sends message $(ID_C, W_C, M_1)$ to server $S$.

`Server S`

1. **Step C10.** Upon receiving the messages, compute $R'_C = W_A \cdot d_s^{-1} = (k'_x, k'_y)$.
2. **Step C11.** Check if $ID'_C = ID_C$, $e(Y'_C, U_C) = e(R'_C, P)$ hold. If the equations do not hold, then stop the session.
3. **Step C12.** Choose $r_s \in Z_q^*$.
4. **Step C13.** Server $S$ sends message $M_2 = R'_C + W_S$, $M_3 = H(W_S)$ to Client $C$.

`Client A`

1. **Step C14.** Compute $W'_S = M_2 - R_C$.
2. **Step C15.** Check if $H(W'_S) = M_3$ holds. If yes, calculate $M4 = H(R'_C, W'_S)$ and send it to Server $S$, else abort.

- $ID_A$: Identity of the client $A$
- $pw_A$: Secret password of the client $A$
- $d_S$ : Secret key of the server $S$
- $U_S$: Public key of the server $S$, where $U_S = d_S \cdot P$
- $r_A / r_S$: Random numbers chosen by the client/server from $\mathbf{Z}_q^*$
- $P$: Basis point of the elliptic group of order $q$, where $q$ is a large prime number
- $e(\cdot)$: Bilinear mapping function.
- $U_A$: Password-verifier of user $A$ , where $U_A = pw_A \cdot P$
- $k_x$: Secret key computed
- $+/-$:Elliptic curve point addition/subtraction
- $H(\cdot)$: collision-resistant hash function
- $E_k(\cdot)$: Symmetric encryption (AES) with key $k$

**Figure 9.** Notations for Hui et al.'s protocol.

The last two phases are omitted since they are not important to the clogging attack demonstration. The protocol claims to be immune against replay attacks; however, we find that it is vulnerable against other attacks, as well.

### 4.5.2. Clogging Attack on Hui et al.'s Protocol

As before, we only need attacker $\mathcal{A}$ to be able to read and modify the contents of messages over an insecure channel (during the login and authentication phase of the protocol).

- $\mathcal{A}$ intercepts a valid login request $(ID_C, W_C, M_1)$ from Step C9
- Since the message is unencrypted, $\mathcal{A}$ changes $ID_C$ to $ID_B$
- $\mathcal{A}$ then sends $(ID_B, W_C, M_1)$ to the server $S$

The server $S$ performs Steps C10 through C15 since the ID matches. $\mathcal{A}$ would now repeat the steps several times and make the server $S$ compute the computationally-intensive elliptic curve bilinear mapping function of Step C11 and collision-resistant hash function of Steps C13, C15 several times. Thus, as in the previous case, the server gets clogged doing unnecessary computations.

Proposed security countermeasures against the clogging attack: The vulnerability of this protocol is due to the fact that the message $(ID_C, W_C, M_1)$ is not encrypted and the timestamp is not used. The protocol can be strengthened by using strong encryption of the messages $M_1$, $M_2$ and $M_3$. Furthermore, adding the timestamp to the messages at the time of run will ensure privacy and thus prevent replay attacks, which leads to clogging.

### 4.5.3. Unauthorized Access Attack

Unauthorized access is the act of gaining access to a network, system, application or other resource without permission. Unauthorized access could occur if a user attempts to access an area of a system they should not be accessing. Unauthorized access could be the result of unmodified default access policies or the lack of clearly-defined access policy documentation [40]. In the paper, [38], $ID_C$ and $U_C$ are stored in a write-protected file in Step $C4$ of the registration phase. Their paper [38] does not mention how the file is stored and also if it is protected using standard security measures. Under the assumption that the $ID_C$ and $U_C$ are unprotected, an unauthorized client could potentially steal the write protected file or eavesdrop on the information exchanged between a legitimate client and the directory server. When the file containing the ID and password becomes accessible to an unauthorized user, the protocol is compromised under this attack.

Proposed security countermeasures against the unauthorized access attack: Unauthorized access can occur from inside the organization or, if the organization is connected to an extranet or to the Internet, from outside. The authentication methods, password policies and access control mechanisms provided by the directory server offer efficient ways of preventing unauthorized access. A network firewall protects a computer network from unauthorized access. Network firewalls may be hardware devices, software programs or a combination of the two. Network firewalls guard an internal computer network against malicious access from the outside. Network firewalls may also be configured to restrict access to the outside from internal users [40].

### 4.5.4. Unauthorized Tampering Attack

Tampering is the unauthorized modification of data by an unauthorized user. When an unauthorized user gains access to the write-protected file in which $ID_C$ and $U_C$ are stored or if they intercept communication between the directory server and a client application, they have the potential to modify the directory data. These unauthorized modifications may include:

- Unauthorized modification of data
- Unauthorized modification of configuration information
- Alteration or cancellation of client's request to the server
- Alteration of the server's response to the client

Adversary $\mathcal{A}$ can cause all other kinds of attacks if the write-protected file is accessible to them as it contains all of the user identity's and passwords.

Proposed security countermeasures against the unauthorized tampering attack: An adversary $\mathcal{A}$ could alter a client's request to the server, cancel the request, or change the server's response to the client. The countermeasures include using the secure socket layer (SSL) protocol to solve this problem by signing information at either end of the connection. Another solution could be to store the $ID_A$ and $U_A$ in the cloud instead of a physical device.

## *4.6. Ammayappan et al.'s Protocol*

The final protocol considered in this work is that of Ammayappan et al. [41]. This is a key agreement protocol and works in a mobile ad hoc network (MANET)-based environment.

Ammayappan et al. show their protocol to be immune to the man-in-the-middle (MITM) attack, and they claim that the protocol's security is based on the ECC logarithm.

### 4.6.1. Review of the Protocol

The protocol by [41] works in two phases: registration phase and active phase. Figure 10 shows the notations used for the protocol.

- $q$: A large prime number
- $AReq$: Authentication Request Packet
- $ARes$: Authentication Response Packet
- $V_K$: Signature verification using key $K$
- $TTP$: Trusted Third Party
- $Token_X$: Hybrid crypt token of node $X$
- $a$: Long term secret of node $A$
- $b$: Long term secret of node $B$
- $r_A$: Ephemeral secret key of node $A$
- $r_B$: Ephemeral secret key of node $B$
- $P$: A base point on elliptic curve
- $Pub_A$: Long term public key of node $A$, $Pub_A = aP$
- $Pub_B$: Long term public key of node $B$, $Pub_B = bP$
- $SK_{AB}$: Session key generated between node $A$ and $B$
- $SK_{BA}$: Session key generated between node $B$ and $A$
- $H(\cdot)$: Secure one way hash function
- $HMAC(\cdot)$: Keyed message authentication code function
- $RN_A$: Random nonce generated by node $A$
- $RN_B$: Random nonce generated by node $B$

**Figure 10.** Notations for Ammayappan et al.'s protocol.

This protocol uses trusted third party (TTP) as a certifying authority. The protocol is presented in Algorithm 8.

### 4.6.2. Man-In-The-Middle Attack on the Protocol

The adversary $E$ has the same power as assumed by Ammayappan et al. in [41] while performing the security analysis of their protocol. $E$ is allowed to read and modify contents of messages over an insecure channel (during the active phase of the protocol). The line of our attack is an MITM attack where the objective of $E$ would be to impersonate one party (e.g., $A$) while communicating with the other (e.g., $B$). E is allowed to read and modify contents of messages over an insecure channel during the active phase of the protocol. Ammayappan et al. claim that their protocol ensures security through the elliptic curve discrete logarithm. However, the protocol can be compromised even with ECC being used in the protocol as a security measure.

**Algorithm 8:** Ammayappan et al.'s protocol.

```
Node A

1. Step A1. Choose a random number $r_A$ and computes $Q_A = r_A \cdot P$ as its public key.
2. Step A2. Node $A$ sends $AReq(Token_A, RN_A, Q_A)$ to Node $B$.

Node B

1. Step B1. Verify $Token_A$.
2. Step B2. Generate a random nonce $RN_B$.
3. Step B3. Select a random integer $r_B$ and compute $Q_B = r_B \cdot P$.
4. Step B4. Compute $SK_{BA} = H((r_B + b) \cdot (Q_A + Pub_A) \| ID_A \| ID_B \| RN_A \| RN_B)$,
   $HMAC_B = H(SK_{BA} \| H((Q_A \cdot x + Q_B \cdot x) \| (Q_A \cdot y + Q_B \cdot y) \| ID_A \| ID_B \| RN_A \| RN_B)$.
5. Step B5. Construct message $m = RN_A \| RN_B \| Q_B \| HMAC_B$.
6. Step B6. Generate $Sig_B(m) = (r, s)$.
7. Step B7. Send $Arep(m, Sig_B(m))$ to Node $A$.

Node A

1. Step A3. Verify $B$'s signature $Sig_B(m)$.
2. Step A4. Verify received $RN_A$ with previous $RN_A$, if no match then
3. Step A5. Compute $SK_{AB} = H((r_A + a) \cdot (Q_B + Pub_B) \| ID_A \| ID_B \| RN_A \| RN_B)$,
   $HMAC_A = H(SK_{AB} \| H((Q_A \cdot x + Q_B \cdot x) \| (Q_A \cdot y + Q_B \cdot y) \| ID_A \| ID_B \| RN_A \| RN_B)$.
4. Step A6. Compare computed $HMAC_A$ with received $HMAC_B$ for integrity check.
5. Step A7. Send an acknowledgment $Sig_A(RN_B \| HMAC_B)$ to $B$.
```

Proposed countermeasures for the man-in-the-middle attack: The vulnerability in this protocol lies in the fact that the mechanism of public key distribution is not mentioned in it, e.g., how node $A$ receives the public key ($Pub_B$) of node $B$ is not mentioned. The attack in Algorithm 9 succeeds if the attacker $E$ manages to make $A$ believe that $Pub_E$ is the public key of $B$. The countermeasure to this attack is to use an integrity check in the messages sent from the TTP to nodes. The following steps may be used to securely distribute (public) keys by the TTP. It is assumed that the TTP has a database storing public keys of all users.

1. When node $A$ wants to communicate to $B$, it sends the following message to the TTP: $A \Rightarrow TTP$: $Sig_A(B \| T_A)$, where $T_A$ is the current timestamp of $A$'s system.
2. TTP upon receipt of the message validates the timestamp.
3. TTP then sends the public key of $B$ to $A$ using the following message: $TTP \Rightarrow A$: $Sig_{TTP}(Pub_B \| T_{TTP} \| H(S))$, where $T_{TTP}$ is the current timestamp of $TTP$'s system, and $S$ is a secret value shared between $A$ and the $TTP$.
4. Upon receipt of the message, $A$ validates the timestamp, recomputes $H(S)$, using the hash and the secret value, and verifies it with the received $H(S)$.

Security analysis of the fix: As mentioned before, the intention of the attacker $E$ would be to somehow send its own public key $Pub_E$ to $A$, making $A$ believe it has the public key of $B$. To achieve this, $E$ could do either of the following:

1. 
   - Change $A$'s request to the $TTP$to $Sig_A(E \| T_A)$.
   - Therefore, have the $TTP$ send back $Sig_{TTP}(Pub_E \| T_{TTP} \| H(S))$ to $A$.
2. Try to change the message $Sig_{TTP}(Pub_B \| T_{TTP} \| H(S))$ to $Sig_{TTP}(Pub_E \| T_{TTP} \| H(S))$.

$E$ would not be able to do the first because $E$ would not know the private key of $A$ needed to create a valid (duplicate) signature of $A$. The timestamp provides protection against a possible replay of an old message by $E$. For the second, since $E$ does not know the secret value $S$, it will not be able to modify the message.

**Algorithm 9:** Ammayappan et al.'s protocol under MITM attack.

```
Node A

1. Step A1. Choose a random number $r_A$ and computes $Q_A = r_A \cdot P$ as its public key.
2. Step A2. Node $A$ sends $AReq(Token_A, RN_A, Q_A)$ to Node $B$.

Adversary E

1. Step E1. Modify the message to $AReq(Token_A, RN_E, Q_E)$ and sends to Node $B$.

Node B

1. Step B1. Compute $SK_{BE} = H((r_B + b) \cdot (Q_E + Pub_E) \| ID_E \| ID_B \| RN_E \| RN_B)$,
   $HMAC_B = H(SK_{BE} \| H((Q_E \cdot x + Q_B \cdot x) \| (Q_E \cdot y + Q_B \cdot y) \| ID_E \| ID_B \| RN_E \| RN_B)$.
2. Step B2. Construct message $m_1 = RN_E \| RN_B \| Q_B \| HMAC_B$.
3. Step B3. Generate $Sig_B(m_1) = (r, s)$.
4. Step B4. Send $Arep(m_1, Sig_B(m_1))$ to Node $A$.

Adversary E

1. Step E2. Intercept $Arep(m_1, Sig_B(m_1))$ sent by $B$ (to $A$).
2. Step E3. Compute $SK_{AE} = H((r_E + e) \cdot (Q_A + Pub_A) \| ID_A \| ID_E \| RN_A \| RN_E)$,
   $HMAC_E = H(SK_{AE} \| H((Q_A \cdot x + Q_E \cdot x) \| (Q_A \cdot y + Q_E \cdot y) \| ID_A \| ID_E \| RN_A \| RN_E)$
3. Step E4. Construct message $m_2 = RN_A || RN_E || Q_E || HMAC_E$
4. Step E5. Generate $Sig_E(m_2) = (r, s)$.
5. Step E6. Send $Arep(m_2, Sig_E(m_2))$ to Node $A$.

Node B

1. Step B5. Compute $SK_{BE} = H((r_E + e).(Q_B + Pub_B) \| ID_E \| ID_B \| RN_E \| RN_B)$,
   $HMAC_E = H(SK_{BE} \| H((Q_E \cdot x + Q_B \cdot x) \| (Q_E \cdot y + Q_B \cdot y) \| ID_E \| ID_B \| RN_E \| RN_B)$.

Node A

1. Step A3. Compute $SK_{AE} = H((r_A + a) \cdot (Q_E + Pub_E) \| ID_A \| ID_E \| RN_A \| RN_E)$,
   $HMAC_A = H(SK_{AE} \| H((Q_A \cdot x + Q_E \cdot x) \| (Q_A \cdot y + Q_E \cdot y) \| ID_A \| ID_E \| RN_A \| RN_E)$.
```

## 5. Summary of the Results

In order to demonstrate the vulnerabilities of the five protocols considered, the author designed generalized cryptanalysis algorithms to perform MITM, database and clogging attacks on the protocols that use ECC as a security measure. First, the protocols of [26,33,36,38] were shown to be vulnerable to the clogging attack. The vulnerability lies in the use of computationally-intensive ECC by the server in the authentication process. In this analysis, it was observed that a combination of encryption, a nonce and a timestamp would prevent clogging attack vulnerability in these three protocols. These protocols were also shown to be vulnerable to database attacks, such as weak authentication, SQL injection, desynchronization, unauthorized access, unauthorized tampering and database protocol vulnerability. In this analysis, it was observed that using input validation, an updated ID process, a network firewall and encryption would prevent database attack vulnerability in these protocols. The key agreement protocol in [41] was analyzed next. Ammayappan et al. showed their protocol in [41] to be immune to the MITM attack, and they claimed that the protocol's security is based on the ECC algorithm. However, in this work, their protocol was shown to be inherently vulnerable to the MITM attack.

Table 2 summarizes the vulnerabilities found in the protocols analyzed. A 'Yes' in a cell indicates that we found the designated protocol vulnerable to the designated attack through a static analysis, and a 'No' means that no vulnerability was found. It is evident that most of the protocols are vulnerable to classical clogging and database attacks.

**Table 2.** Summary of the vulnerabilities.

| Protocol | Clogging | MITM | Weak Authentication | SQL Injection | Unauth Access | DB Vulnerability | Desynch | Unauth Tampering |
|---|---|---|---|---|---|---|---|---|
| Moosavi et al. [33] | Yes | No | No | No | No | No | No | No |
| Xu et al. [26] | Yes | No | Yes | Yes | Yes | No | No | No |
| Xu et al. [36] | Yes | No | Yes | No | No | Yes | Yes | No |
| Hui et al. [38] | Yes | No | No | No | Yes | No | No | Yes |
| Ammayappan et al. [41] | No | Yes | No | No | No | No | No | No |

The countermeasures are also summarized in Table 3. It must be emphasized that as everything has costs involved, the level of security needed will determine the nature of the counter-measure.

**Table 3.** Summary of the results.

| Protocol | Mode of Attack | Countermeasure |
|---|---|---|
| Moosavi et al. [33] | Classical Clogging | Validity Checks |
| Xu et al. [26] | Classical Clogging, Weak Authentication, SQL Injection, Unauthorized Access | Efficient Algorithm, input validation, Timestamp |
| He et al. [36] | Classical Clogging, Weak Authentication, Database Protocol Vulnerability, Desynchronization | Input Validation, Updating of ID, Encryption, Timestamp |
| Hui et al. [38] | Classical Clogging, Unauthorized Access, Unauthorized Tampering | Usage of SSL, Timestamp, Network firewalls |
| Ammayappan et al. [41] | Man in the Middle | Securing Public Key Management |

## 6. Conclusions

In this work, clogging attacks and database attacks have been demonstrated on five advanced ECC-based authentication schemes. The goal was to uncover the subtleties and challenges in designing such protocols. ECC techniques guarantee a level of security. However, they might still contain an easily-exploitable vulnerability if they are used without an additional level of protection. Hence, some level of protection should be added to them to guarantee total security against clogging attacks.

It is concluded that ECC guarantees a level of security, but it might create a vulnerability if it is used without an additional level of protection. Most of the multi-factor authentication and key exchange protocols in the literature, whether smart card or RFID based, rely on ECC for their security. Hence, an additional layer of defense should be added to them to guarantee increased security against MITM, database and clogging attacks.

### *Directions of Future Research*

Research on elliptic curve cryptography authentication protocols is ongoing. This research has been conducted using static analysis. An obvious next step is to test the dynamic vulnerability of these protocols. The documentation presented here will lay the groundwork for performing dynamic analysis on ECC-based protocols. This documentation provides detailed steps and procedures to perform static analysis. Our suggested countermeasures on strengthening the security flaws in the above protocols will provide a strong basis to perform dynamic analysis. Other kinds of security flaws could potentially be discovered while performing dynamic analysis.

**Acknowledgments:** The research work presented in this article was supported in part by a seed grant awarded by the University of North Florida to the first author. The second author was supported as a research assistant from that grant. The authors thank the two anonymous reviewers for improving the quality of presentation of this article.

**Author Contributions:** The ideas of cryptanalysis of the illustrated protocols in this work was conceived by S.R. (the first author). C.K. (the second author) carried out the statical analysis to establish these ideas. The manuscript was written mostly by S.R.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Khatwani, C.; Roy, S. Security Analysis of ECC Based Authentication Protocols. In Proceedings of the 2015 International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, India, 12–14 December 2015; pp. 1167–1172.
2. Burr, J. Elliptical Curve Cryptography (ECC). Available online: http://searchsecurity.techtarget.com/definition/elliptical-curve-cryptography/ (accessed on 19 July 2016).
3. Abidi, A.; Bouallegue, B.; Kahri, F. Implementation of elliptic curve digital signature algorithm (ECDSA). In Proceedings of the 2014 Global Summit on Computer & Information Technology (GSCIT), Sousse, Tunisia, 14–16 June 2014; pp. 1–6.
4. Choi, Y.; Lee, D.; Kim, J.; Jung, J.; Nam, J.; Won, D. Security enhanced user authentication protocol for wireless sensor networks using elliptic curves cryptography. *Sensors* **2014**, *14*, 10081–10106.
5. Chuang, Y.H.; Hsu, C.L.; Shu, W.; Hsu, K.C.; Liao, M.W. A Secure Non-interactive Deniable Authentication Protocol with Certificates Based on Elliptic Curve Cryptography. In *New Trends in Intelligent Information and Database Systems*; Springer: Berlin, Germany, 2015; pp. 183–190.
6. Jin, C.; Xu, C.; Zhang, X.; Zhao, J. A secure RFID mutual authentication protocol for healthcare environments using elliptic curve cryptography. *J. Med. Syst.* **2015**, *39*, 1–8.
7. Yeh, H.L.; Chen, T.H.; Shih, W.K. Robust smart card secured authentication scheme on SIP using elliptic curve cryptography. *Comput. Stand. Interfaces* **2014**, *36*, 397–402.
8. Zhang, L.; Tang, S.; Chen, J.; Zhu, S. Two-Factor Remote Authentication Protocol with User Anonymity Based on Elliptic Curve Cryptography. *Wirel. Pers. Commun.* **2015**, *81*, 53–75.
9. Chande, M.K.; Lee, C.C. An improvement of a elliptic curve digital signature algorithm. *Int. J. Internet Technol. Secur. Trans.* **2016**, *6*, 219–230.
10. Hwang, M.S.; Lee, C.C.; Lee, J.Z.; Yang, C.C. A secure protocol for bluetooth piconets using elliptic curve cryptography. *Telecommun. Syst.* **2005**, *29*, 165–180.
11. Lee, C.C.; Li, C.T.; Weng, C.Y.; Jheng, J.J.; Zhang, X.Q.; Zhu, Y.R. *Cryptanalysis and Improvement of an ECC-Based Password Authentication Scheme Using Smart Cards*; CSS. Springer: Milpitas, CA, USA, 2013; pp. 338–348.
12. Lo, J.W.; Lee, C.C.; Hwang, M.S.; Chu, Y.P. A secure and efficient ECC-based AKA protocol for wireless mobile communications. *Int. J. Innov. Comput. Inf. Control* **2010**, *6*, 5249–5258.
13. Chande, M.K.; Li, C.T.; Lee, C.C. A CAE Scheme Using ECC Based Self Certified PKC. *J. Comput. Sci.* **2016**, *12*, 527–533.
14. Guide, R. Elliptical Curve Cryptography (ECC). Available online: http://www.sysax.com/ftblog/windows-ftp/elliptic-curve-cryptography-ecc/ (accessed on 19 July 2016).
15. Farash, M.S.; Ahmadian-Attari, M. A Pairing-Free ID-Based Key Agreement Protocol with Different PKGs. *IJ Netw. Secur.* **2014**, *16*, 143–148.
16. Bhargav-Spantzel, A.; Squicciarini, A.C.; Modi, S.; Young, M.; Bertino, E.; Elliott, S.J. Privacy preserving multi-factor authentication with biometrics. *J. Comput. Secur.* **2007**, *15*, 529–560.
17. Owen, W.N.; Shoemaker, E. Multi-Factor Authentication System. U.S. Patent 7,373,515, 13 May 2008.
18. Sabzevar, A.P.; Stavrou, A. Universal multi-factor authentication using graphical passwords. In Proceedings of the 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems, Bali, Indonesia, 30 November–3 December 2008; pp. 625–632.

19. Chang, R.; Jiang, G.; Ivancic, F.; Sankaranarayanan, S.; Shmatikov, V. Inputs of coma: Static detection of denial-of-service vulnerabilities. In Proceedings of the 2009 22nd IEEE Computer Security Foundations Symposium, Port Jefferson, NY, USA, 8–10 July 2009; pp. 186–199.
20. Sullivan, N. A (Relatively Easy to Understand) Primer on Elliptic Curve Cryptography. Available online: https://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/ (accessed on 24 October 2013).
21. Hankerson, D.; Menezes, A.J.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer Science & Business Media: Berlin, Germany, 2006.
22. Graham, J.; Olson, R.; Howard, R. *Cyber Security Essentials*; CRC Press: Boca Raton, FL, USA, 2016.
23. Koblitz, N. Elliptic curve cryptosystems. *Math. Comput.* **1987**, *48*, 203–209.
24. Koc, C.K. Elliptic Curve Cryptography. Available online: https://koclab.cs.ucsb.edu/ (accessed on 2 February 2017).
25. Garrett, K.; Talluri, S.R.; Roy, S. On vulnerability analysis of several password authentication protocols. *Innov. Syst. Softw. Eng.* **2015**, *11*, 167–176.
26. Xu, L.; Wu, F. An improved and provable remote user authentication scheme based on elliptic curve cryptosystem with user anonymity. *Secur. Commun. Netw.* **2015**, *8*, 245–260.
27. Higgins, K.J. Hacker's Choice: Top Six Database Attacks. Available online: http://www.darkreading.com/risk/hackers-choice-top-six-database-attacks/d/d-id/1129481?/ (accessed on 19 July 2016).
28. Winkler, D.C. Securing Your Password Database with Bcrypt. Available online: https://en.wikipedia.org/wiki/Elliptic_curve_Diffie-Hellman/ (accessed on 19 July 2016).
29. Wikipedia. Elliptic Curve Diffie-Hellman. Available online: http://blog.mgm-tp.com/2013/02/securing-your-password-database-using-bcrypt/ (accessed on 19 July 2016).
30. LaMacchia, B.A.; Manferdelli, J.L. New Vistas in elliptic curve cryptography. *Inf. Secur. Tech. Rep.* **2006**, *11*, 186–192.
31. Bos, J.; Kaihara, M.; Kleinjung, T.; Lenstra, A.K.; Montgomery, P.L. *On the Security of 1024-Bit RSA and 160-Bit Elliptic Curve Cryptography*; EPFL-REPORT-164549, 2009.
32. Sherwood, T.; Irvine, C.; Huffmire, T.; Levin, T.; Valamehr, J.; Kaya Koc, C.; Kastner, R. A Qualitative Security Analysis of a New Class of 3-D Integrated Crypto Co-processors. In *Cryptography and Security: From Theory to Applications*; Springer Verlag GmbH: Heidelberg, Germany, 2012.
33. Moosavi, S.R.; Nigussie, E.; Virtanen, S.; Isoaho, J. An elliptic curve-based mutual authentication scheme for RFID implant systems. *Procedia Comput. Sci.* **2014**, *32*, 198–206.
34. Orman, H. *The OAKLEY Key Determination Protocol*; Technical Report; University of Arizona Tucson: Tucson, AZ, USA, 1998.
35. Li, C.T. A new password authentication and user anonymity scheme based on elliptic curve cryptography and smart card. *IET Inf. Secur.* **2013**, *7*, 3–10.
36. He, D.; Kumar, N.; Chilamkurti, N.; Lee, J.H. Lightweight ECC based RFID authentication integrated with an ID verifier transfer protocol. *J. Med. Syst.* **2014**, *38*, 1–6.
37. Liao, Y.P.; Hsiao, C.M. A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol. *Ad Hoc Netw.* **2014**, *18*, 133–146.
38. Wang, S.-H.; Chang, S.-Q.; Wang, Z.W.; Sun, G.Z. A Password Authentication and Update Scheme Based on Elliptic Curve Cryptography. *Int. J. Adv. Comput. Technol.* **2012**, *4*, 84–90.
39. Islam, S.H.; Biswas, G. Design of improved password authentication and update scheme based on elliptic curve cryptography. *Math. Comput. Model.* **2013**, *57*, 2703–2717.
40. Telelink Telecommunication Services Ltd. Unauthorized Access Attack. Available online: http://itsecurity.telelink.com/unauthorized-access-attack/ (accessed on 19 July 2016).
41. Ammayappan, K.; Negi, A.; Sastry, V.; Das, A.K. An ECC-based two-party authenticated key agreement protocol for mobile ad hoc networks. *J. Comput.* **2011**, *6*, 2408–2416.