mathematics

MDPI

Article

Cryptanalysis of Two Privacy-Preserving Authentication Schemes for Smart Healthcare Applications

Feihong Xu [1,*], Junwei Luo [2] and Rahman Ziaur [3]

[1] School of Artificial Intelligence, Wuchang University of Technology, Wuhan 430223, China
[2] School of Computing Technologies, RMIT University, Melbourne, VIC 3083, Australia; s3616926@student.rmit.edu.au
[3] School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia; rahman.ziaur@qut.edu.au
* Correspondence: 120160287@wut.edu.cn

Abstract: Ensuring the secure sharing of privacy-sensitive healthcare data is attracting considerable interest from researchers. Recently, Ogundoyin et al. designed a lightweight privacy-preserving authentication scheme named PAASH for smart health applications. Benil et al. proposed a public verification and auditing scheme named ECACS for securing e-health systems. Ogundoyin et al. and Benil et al. proposed an efficient certificateless aggregate signature (CLAS) scheme as their respective foundation signature schemes. They declared that their constructions were provably secure under the hardness assumption of cryptographic problems. In this work, we disprove their claim by analyzing the correctness and security of their underlying CLAS schemes. We first show that the batch verification process of n signatures for the CLAS scheme in PAASH is incorrect, and any public-key replacement attacker can easily break the scheme. We analyze the reasons for our attack and propose an improved scheme, named PAASH$^+$. We then show that the CLAS scheme in ECACS fails to achieve correctness, an essential property that a cryptographic scheme should provide. As a result, it is impractical to deploy the designed PAASH and ECACS constructions in any real smart health applications.

Keywords: IoT; smart health; electronic health records; privacy-preserving; certificateless signature

MSC: 94A62

Citation: Xu, F.; Luo, J.; Ziaur, R. Cryptanalysis of Two Privacy-Preserving Authentication Schemes for Smart Healthcare Applications. *Mathematics* **2023**, *11*, 3314. https://doi.org/10.3390/math11153314

Academic Editor: Hui Cui

Received: 20 June 2023
Revised: 26 July 2023
Accepted: 27 July 2023
Published: 28 July 2023

1. Introduction

Nowadays, including intelligent transportation, smart industry, and smart healthcare, Internet of Things (IoT) technologies are widely used in various fields of our life. The market research company IDC estimates that 41.6 billion IoT devices will be connected by 2025 [1]. Taking the healthcare industry as an example, smart devices (e.g., smartphones, wearable or implantable sensors) are deployed in applications such as telemedicine, patient physiological monitoring, and clinical problem identification [2]. They collaborate with wireless communication technologies to make health data collection increasingly flexible. Due to the resource limitation of these devices, the collected health data will typically be outsourced to a remote cloud for storage and sharing. However, because of the openness of the network, these privacy-sensitive healthcare data are vulnerable to various security attacks such as eavesdropping and tampering, posing a serious threat to patients' health. Therefore, it is essential to design effective mechanisms to ensure data security and privacy [3].

Historically, digital signatures have typically been used to ensure data authenticity and integrity. However, given the large amount of healthcare data that needs to be received from various smart devices, signature verification must be remarkably efficient to support low-latency emergencies. Consequently, the concept of aggregate signature

introduced by Boneh et al. [4] is an ideal signature technique for supporting batch verification. Boneh et al.'s construction is based on the traditional public-key cryptosystem (PKC), which suffers from the costly key management problem. In view of this, aggregate signature schemes [5] based on an identity-based cryptosystem (IBC) have been proposed; nevertheless, Al-Riyami et al. [6] showed that IBC-based construction has the key escrow problem. To reduce the certificate management cost in traditional PKCs and eliminate the key escrow problem in IBCs, aggregate signature schemes based on the certificateless cryptosystem [7–12] have been put forward for smart healthcare applications.

Recently, Zhang et al. [13] integrated the ciphertext-policy attribute-based encryption (CP-ABE) technique and the certificateless aggregate signature (CLAS) scheme to design a secure smart health system called SSH. Notably, in a CP-ABE scheme, the data owner can apply an attribute secret key based on a set of attributes. The ciphertext is produced under an access policy, and decryption succeeds only when the owner's attribute list satisfies the policy. Therefore, Zhang et al.'s SSH construction simultaneously achieves aggregate authentication, fine-grained access control, and data confidentiality. However, very recently, Ogundoyin et al. [2] found that Zhang et al.'s SSH scheme is vulnerable to any Type II attacker who knows the system master-secret key. To this end, Ogundoyin et al. put forward a new lightweight privacy-preserving authentication scheme named PAASH for smart health applications, which also consists of a CP-ABE scheme and a CLAS scheme. Ogundoyin et al. presented a formal security analysis of their design under the standard cryptographic assumption. To achieve aggregate authentication and data confidentiality, there are also several certificateless signcryption schemes [14,15] in the literature. However, they cannot achieve fine-grained access control.

In addition, as another important smart healthcare application, an electronic health record (EHR) is a record in an electronic version containing personal health-related data, which is stored and retrieved by different healthcare providers for healthcare-related purposes. Compared to the traditional paper-based health record, the EHR has many benefits, such as lowering costs, improving health care quality, promoting evidence-based medicine usage, helping in record-keeping, and ensuring the records' mobility. To secure the storage and sharing of privacy-sensitive EHR data over the potentially untrusted cloud, many cryptographic schemes have been proposed in the literature [16–18]. Zhu et al. [16] proposed an authentication scheme with privacy protection and designated verification for XML-based healthcare records. Domadiya et al. [17] put forward a privacy-preserving scheme for distributed healthcare data collection and mining for EHR systems, which provides a healthcare data mining platform to medical researchers and physicians. Guo et al. [18] presented a hybrid blockchain-edge architecture for managing EHR with attribute-based cryptographic mechanisms. However, these designs require expensive pairing operations. Recently, Benil et al. [19] proposed a new public verification and auditing scheme called ECACS to secure EHRs. ECACS combines a certificateless public verification (CPV) scheme and a certificateless public auditing (CPA) scheme. Designed based on a CLAS scheme, the CPV scheme is used to guarantee the authenticity of the shared EHR data. Moreover, in combination with blockchain technology, the CPA scheme allows a cloud server to generate proof of possession of the EHR data and an auditor to check the correctness of the proof (please refer to [19] for the detailed application model).

Contributions. In this work, we observe that Ogundoyin et al.'s PAASH in [2] cannot satisfy correctness and unforgeability. More specifically, we find that the batch verification process of n signatures for the CLAS scheme in PAASH is incorrect, and any Type I attacker (public-key replacement attacker) can easily break the scheme. Namely, the attacker can impersonate a data owner (i.e., a patient) to forge a valid signature on his false healthcare information. This may lead healthcare professionals to make incorrect diagnoses of a patient's health condition, even with catastrophic consequences. Therefore, the designed PAASH cannot be deployed in practical smart health applications. Based on our cryptanalysis, we discuss the related reason. Moreover, we suggest an improved PAASH$^+$ scheme to fill this gap.

In addition, we show that the CLAS scheme in Benil et al.'s ECACS [19] fails to achieve correctness, the essential property that a cryptographic scheme should provide. Without correctness, other properties such as security and privacy become unattainable. As a result, we demonstrate that their scheme is incorrect/insecure for practical applications.

Organization. The remaining paper is organized as follows: We review some preliminaries in Section 2. In Section 3, we show our cryptanalysis to Ogundoyin et al.'s PAASH scheme. More concretely, we review their scheme in Section 3.1, provide the scheme analysis in Section 3.2, and present our discussion and improvement in Section 3.3. In Section 4, we show our cryptanalysis of Benil et al.'s ECACS Scheme, including a scheme review in Section 4.1 and a scheme analysis in Section 4.2. Conclusions are drawn in Section 5.

2. Preliminaries

We here introduce some preliminaries.

Notations. We use $\mathbb{N} = \{1, 2, \ldots\}$ and $||$ to represent the set of positive integers and the concatenation of two strings, respectively. Denote by $x \in_R X$ the element x chosen uniformly and randomly from the set X.

Elliptic curve discrete logarithm problem (ECDLP) [20]. Given an elliptic curve E defined over a finite field $\mathbb{F}_p$, an additive cyclic subgroup $\mathbb{G}$ on $E/\mathbb{F}_p$ with generator P and prime order q, and a point $Q \in \mathbb{G}$, the ECDLP is to find $\alpha \in \mathbb{Z}_q^*$ such that $Q = \alpha P$.

Correspondingly, the ECDLP assumption refers to any probabilistic polynomial time (p.p.t) adversary $\mathcal{A}$ with negligible probability in solving the ECDLP.

3. Cryptanalysis of the PAASH Scheme in [2]

3.1. Review of the PAASH Scheme

The PAASH scheme is as follows; some notations for the scheme are shown in Table 1.

Table 1. Notations and descriptions.

Notations	Descriptions
ℓ	System security parameter
pa_{ABE}	Attribute-based parameter
x_{ABE}	Attribute-based master secret key
$pparam$	System public parameter
H_i	Hash function, $i = 1, 2, 3$
$(ID_i, {L_i}^{att})$	Patient P_i's identity and attribute list
${SK_i}^{att}$	P_i's attribute-based secret key
PSK_i	P_i's partial secret key
(SK_i, PK_i)	(Full) secret key and public key of P_i
m_i	P_i's healthcare data
c_{po_i}	Ciphertext of m_i under ${SK_i}^{att}$
σ_i	Signature of c_{po_i}
σ	Aggregate signature

- System Setup Phase

  Taking a security parameter ℓ as an input, the registration authority (RA) runs the underlying CP-ABE scheme in [21], which consists of the ABE.MasterKeyGen, ABE.AttributeKeyGen, ABE.Encrypt, and ABE.Decrypt algorithms. More concretely, it runs ABE.MasterKeyGen to obtain an attribute-based parameter pa_{ABE} and an attribute-based master secret key x_{ABE}. Then it executes the following algorithm MasterKenGen to output a public parameter $param$ and a master secret key s.

  MasterKenGen: the RA performs the following:

  1. Choose a non-singular elliptic curve $E : y^2 = x^3 + ax + b \mod p$ and a q-order additive group $\mathbb{G} = \langle P \rangle$, where p, q are large primes.

2. Pick $s \in_R \mathbb{Z}_q^*$ as the master secret key and compute the corresponding public key $P_{pub} = sP$.
3. Select three hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$, $H_3 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$.
4. Store s secretly and broadcast public parameters $pparam = \{E, \mathbb{G}, p, q, P, P_{pub}, H_1, H_2, H_3\}$.

- Registration Phase

Let a patient P_i be the smart health client with identity ID_i and an attribute list ${L_i}^{att}$. When P_i wants to join the system, P_i does the following:

1. Choose $d_i \in_R \mathbb{Z}_q^*$ and a timestamp ${t_i}^1$.
2. Compute $D_i = d_i P$, $Y_i = d_i P_{pub}$, and $\Omega_i = ID_i || {L_i}^{att} \bigoplus H_1(Y_i || {t_i}^1)$.
3. Send $(\Omega_i, D_i, {t_i}^1)$ to the RA as its registration request.

Upon receiving $(\Omega_i, D_i, {t_i}^1)$ from P_i, the RA operates as follows:

1. Accept the request if ${t_i}^1$ is fresh and reject it otherwise.
2. Recover ID_i and ${L_i}^{att}$ by computing $ID_i || {L_i}^{att} = \Omega_i \bigoplus H_1(sD_i || {t_i}^1)$.
3. Confirm the validity of ID_i and compute $PID_i = H_1(s || ID_i || T_i)$ as the pseudonym of P_i, where T_i is the valid time period.

Now, the RA runs ABE.AttributeKeyGen (i.e., using x_{ABE}) to generate P_i's attribute-based secret key ${SK_i}^{att}$ and executes the following PartialSecKeyGen to generate P_i's partial secret key PSK_i. Notably, P_i will use ${SK_i}^{att}$ and PSK_i to encrypt the healthcare data and sign the corresponding ciphertext, respectively.
PartialSecKeyGen: the RA performs the following:

1. Choose a timestamp ${t_R}^1$.
2. Compute $\alpha_i = H_2(ID_i || s || P_{pub} || T_i)$, $\beta_i = H_2(ID_i || {L_i}^{att} || pparam)$, $\theta_i = \alpha_i \beta_i + s$, and $A_i = \alpha_i P$.
3. Set $PSK_i = (A_i, \theta_i)$ as the partial secret key.
4. Compute a masking parameter $\Theta_i = H_1(sD_i || {t_R}^1) \bigoplus ({SK_i}^{att} || PSK_i)$.
5. Send $(PID_i, \Theta_i, {t_R}^1)$ to P_i.

Upon receiving $(PID_i, \Theta_i, {t_R}^1)$, P_i operates as follows:

1. Check the freshness of ${t_R}^1$.
2. Compute ${SK_i}^{att} || PSK_i = \Theta_i \bigoplus H_1(sD_i || {t_R}^1)$ to recover ${SK_i}^{att}$ and PSK_i. The validity of these keys can be verified by $\theta_i P = \beta_i A_i + P_{pub}$, where $\beta_i = H_2(ID_i || {L_i}^{att} || pparam)$.

- Data Outsourcing Phase

P_i with pseudonym PID_i outsources its healthcare data m_i to a medical cloud server (MCS) where authorized data requesters (e.g., healthcare providers or researchers) can access it. To do this, P_i firstly defines a ciphertext policy po_i and executes ABE.Encrypt to obtain the ciphertext c_{po_i} corresponding to m_i. Then, P_i executes the following algorithms SecretKeyGen and Sign:
SecretKeyGen: P_i picks $x_i, y_i \in_R \mathbb{Z}_q^*$ and computes $PK_i = y_i P$. Now, P_i sets $SK_i = (x_i, y_i, \theta_i)$ as the secret key and PK_i as the public key.
Sign: P_i does the following:

1. Select $r_i \in_R \mathbb{Z}_q^*$ and compute $V_i = r_i x_i P$, $R_i = V_i + \beta_i A_i$, $h_i = H_3(c_{po_i} || PID_i || R_i || PK_i || {t_i}^2)$, and $\delta_i = r_i x_i + h_i y_i + \theta_i$, where ${t_i}^2$ is a timestamp and $\beta_i = H_2(ID_i || {L_i}^{att} || pparam)$.
2. Set $\sigma_i = (R_i, \delta_i)$ as a signature on $c_{po_i} || {t_i}^2$.

P_i now uploads the tuple $(c_{po_i}, PID_i, \sigma_i, PK_i, {t_i}^2)$ to the MCS.
Suppose that the MCS receives a set of $(c_{po_i}, PID_i, \sigma_i, PK_i, {t_i}^2)$ from P_i for $i = 1, 2, \ldots, n$. It executes the algorithms Aggregate and AggregateVerify:
Aggregate: the MCS performs the following:

1. Check the freshness of ${t_i}^2$ and recover $h_i = H_3(c_{po_i}||PID_i||R_i||PK_i||{t_i}^2)$ for $i = 1, 2, \ldots, n$.
2. Calculate $\Delta = \Sigma_{i=1}^{n} \delta_i$, $R = \Sigma_{i=1}^{n} R_i$, and $PK = \Sigma_{i=1}^{n} h_i PK_i$.
3. Return $\sigma = (R, \Delta)$ as the aggregate signature on $c_{po_i}||{t_i}^2$, $i = 1, 2, \ldots, n$.

AggregateVerify: The MCS accepts the message $c_{po_i}||{t_i}^2$, $i = 1, 2, \ldots, n$ if $\Delta P = R + PK + P_{pub}$ holds and rejects otherwise.

- Data Access Phase

A data requester (e.g., a healthcare provider or a researcher) is allowed to download P_i's healthcare information $(c_{po_i}, PID_i, \sigma_i = (R_i, \delta_i), PK_i, {t_i}^2)$ from the MCS. More concretely, it firstly executes the following Verify algorithm:

Verify: the data requester checks the freshness of ${t_i}^2$, recovers $h_i = H_3(c_{po_i}||PID_i||R_i||PK_i||{t_i}^2)$, and verifies whether $\delta_i P = R_i + h_i PK_i + P_{pub}$. Note that the data requester can also check a set of messages simultaneously as in AggregateVerify to improve verification efficiency.

At last, if the data requester's attributes match with the access policy po_i, it can run the ABE.Decrypt to obtain the healthcare data m_i.

3.2. Scheme Analysis

The above PAASH scheme proposed by Ogundoyin et al. can be seen as a combination of two cryptographic schemes, i.e., the underlying lightweight CP-ABE scheme proposed by Gafif et al. in [21] and a new CLAS scheme designed by Ogundoyin et al. Here, the CLAS scheme is constructed by algorithms MasterKenGen, PartialSecretKeyGen, SecretKeyGen, Sign, Verify, Aggregate, and AggregateVerify. Meanwhile, the first five algorithms in CLAS naturally form a basic CLS scheme. Since the security of the CP-ABE scheme has been analyzed by Gafif et al., we only focus on the correctness and security analysis of Ogundoyin et al.'s CLAS scheme.

3.2.1. Correctness Analysis

In Ogundoyin et al.'s CLS scheme, upon obtaining P_i's healthcare information $(c_{po_i}, PID_i, \sigma_i = (R_i, \delta_i), PK_i, {t_i}^2)$, the verifier can use Verify to check the single signature σ_i by checking whether the equation $\delta_i P = R_i + h_i PK_i + P_{pub}$ holds (we hereafter omit the hash operation in correctness analysis). This is true since the verification equation can easily verify a single valid signature. However, their aggregate signature on n signatures produced by Aggregate cannot be validated by AggregateVerify.

For example, when $n = 2$, the MCS in Aggregate computes $\Delta = \delta_1 + \delta_2$, $R = R_1 + R_2$, and $PK = h_1 PK_1 + h_2 PK_2$. It then sets $\sigma = (R, \Delta)$ as the aggregate signature on $c_{po_1}||{t_1}^2$ and $c_{po_2}||{t_2}^2$. To accept the signature σ, in AggregateVerify, one needs to check whether $\Delta P = R + PK + P_{pub}$ holds. However, it is obvious that the equation does not hold since

$$\begin{aligned} \Delta P =& (\delta_1 + \delta_2) P \\ =& R_1 + h_1 PK_1 + 2P_{pub} + R_2 + h_2 PK_2 \\ =& R + PK + 2P_{pub} \\ \neq& R + PK + P_{pub}. \end{aligned}$$

To fix the problem, the verification equation in AggregateVerify needs to be changed to $\Delta P = R + PK + nP_{pub}$.

In the following section, we show that the fixed CLAS scheme is insecure.

3.2.2. Security Analysis

In [2], Ogundoyin et al. proved that their scheme can resist two types of security attacks. Namely, the Type I adversary is regarded as a public key replacement attacker. It knows the secret value of a target user but does not know the user's partial secret key. Meanwhile, the Type II adversary is regarded as a malicious-but-passive KGC who is able to

know the master secret key but does not obtain the secret value of the target user. We refer the readers to [22,23] for the detailed security model. Here, we disprove Ogundoyin et al.'s claim by presenting a concrete Type I attack.

We suppose that $\mathcal{A}_1$ is a Type I adversary, as shown in Figure 1. $\mathcal{A}_1$'s goal is to forge a signature $\sigma_i^* = (R_i^*, \delta_i^*)$ on a message $c_{po_i}^* || t_i^2$ for a target participant P_i with the pseudonym PID_i, where t_i^2 is a random timestamp. Now, $\mathcal{A}_1$ will be provided with the system parameter $pparam = \{E, \mathbb{G}, p, q, P, P_{pub}, H_1, H_2, H_3\}$ and P_i's public information (PID_i, PK_i).

$\mathcal{A}_1$ picks $y_i^* \in_R \mathbb{Z}_q^*$, computes $PK_i^* = y_i^* P$, and sets PK_i^* as the replaced public key. $\mathcal{A}_1$ now executes as follows:

1. Select $z \in_R \mathbb{Z}_q^*$ and set $R_i^* = zP - P_{pub}$.
2. Compute $h_i^* = H_3(c_{po_i}^* || PID_i || R_i^* || PK_i^* || t_i^2)$ and $\delta_i^* = z + h_i^* y_i^*$.
3. Set $\sigma_i^* = (R_i^*, \delta_i^*)$ as its forgery.

$\mathcal{A}_1$ now sends the tuple $(c_{po_i}^*, PID_i, \sigma_i^*, PK_i^*, t_i^2)$ to a potential verifier (i.e., the MCS or the data requester). The correctness of $\sigma_i^* = (R_i^*, \delta_i^*)$ is verified as:

$$\begin{aligned} \delta_i^* P =& (z + h_i^* y_i^*) P = zP + h_i^* y_i^* P \\ =& R_i^* + P_{pub} + h_i^* PK_i^* \\ =& R_i^* + H_3(c_{po_i}^* || PID_i || R_i^* || PK_i^* || t_i^2) PK_i^* + P_{pub}. \end{aligned}$$

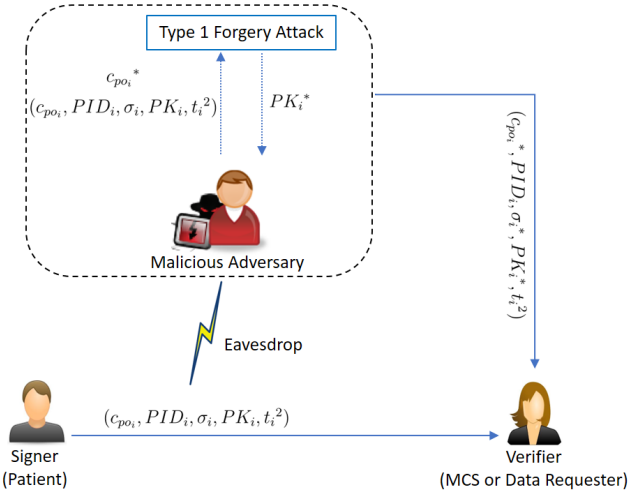

Figure 1. An example of the Type I attack.

Apparently, $\mathcal{A}_1$ is successful in forging the signature on $c_{po_i}^* || t_i^2$. Since any single signature can be forged, the designed CLAS cannot provide its claimed security assurance.

Recall that in Ogundoyin et al.'s scheme, P_i encrypts its healthcare data m_i by running ABE.Encrypt. One may argue that the described adversary $\mathcal{A}_1$ cannot arbitrarily forge ciphertext c_{po_i} since it does not know the attribute-based secret key SK_i^{att} of P_i. However, even with this, $\mathcal{A}_1$ is still able to accomplish its forgery by freely selecting the $c_{po_i}^*$ from the historical encrypted data sent by P_i. Note that data requesters need to provide medical services such as diagnosis and treatment to the patient P_i based on the received $c_{po_i}^*$. However, incorrect or untimely $c_{po_i}^*$ may lead to misdiagnosis of a patient's health condition and, in severe cases, may even pose a significant health risk to the patient. Therefore, the proposed PAASH scheme cannot be deployed to real applications.

3.3. Discussion and Improvement

In Ogundoyin et al.'s scheme, R_i and P_{pub} are independent of each other in the verification equation $\delta_i P = R_i + h_i PK_i + P_{pub}$. This allows $\mathcal{A}_1$ to use the algebraic relationship

between them to generate an intermediate value that can be used to bypass the master secret key s and thus forge the signature on c_{po_i}.

In view of this, we suggested an improved PAASH$^+$ scheme as below. Note that our improvement mainly focuses on their proposed CLAS scheme.

Recall that in the System Setup Phase, the RA in MasterKeyGen selects three hash functions. Here, we let RA choose four hash functions $H_i : \{0,1\}^* \to \mathbb{Z}_q^*$, $i = 1,2,3,4$. The remaining part of the algorithm is the same as the original algorithm.

In the Registration Phase, the RA sets $A_i = \alpha_i \beta_i P$. As a result, the partial secret key $PSK_i = (A_i, \theta_i)$ can be checked by $\theta_i P = A_i + P_{pub}$. The remaining part of the PartialSecretKeyGen algorithm is the same as the original algorithm.

In the Data Outsourcing Phase, four algorithms—SecretKeyGen, Sign, Aggregate, and AggregateVerify—are changed as follows:

SecretKeyGen: P_i picks $x_i \in_R \mathbb{Z}_q^*$ and computes $X_i = x_i P$. Now, P_i sets $SK_i = (x_i, \theta_i)$ as the secret key and $PK_i = (A_i, X_i)$ as the public key.

Sign: P_i works as follows:

1. Pick $r_i \in_R \mathbb{Z}_q^*$ and compute $R_i = r_i P, h_{3i} = H_3(PID_i||PK_i||P_{pub}), h_{4i} = H_4(c_{po_i}||PID_i||R_i||PK_i||{t_i}^2)$, and $\delta_i = \theta_i + r_i h_{4i} + x_i h_{3i}$, where ${t_i}^2$ is a timestamp.
2. Set $\sigma_i = (R_i, \delta_i)$ as a signature on $c_{po_i}||{t_i}^2$.

P_i now uploads the tuple $(c_{po_i}, PID_i, \sigma_i, PK_i, {t_i}^2)$ to the MCS.

Suppose that the MCS receives a set of $(c_{po_i}, PID_i, \sigma_i, PK_i, {t_i}^2)$ from P_i for $i = 1,2,\ldots,n$. It executes the algorithms Aggregate and AggregateVerify:

Aggregate: the MCS performs the following:

1. Check the freshness of ${t_i}^2$ and calculate $\Delta = \Sigma_{i=1}^n \delta_i$.
2. Return $\sigma = (\Delta, R_1, R_2, \ldots, R_n)$ as the aggregate signature on $c_{po_i}||{t_i}^2$, $i = 1,2,\ldots,n$.

AggregateVerify: the MCS accepts the message $c_{po_i}||{t_i}^2$, $i = 1,2,\ldots,n$ if $\Delta P = \sum_{i=1}^n A_i + nP_{pub} + \sum_{i=1}^n h_{3i}X_i + \sum_{i=1}^n h_{4i}R_i$ holds and rejects otherwise, where $h_{3i} = H_3(PID_i||PK_i||P_{pub})$ and $h_{4i} = H_4(c_{po_i}||PID_i||R_i||PK_i||{t_i}^2)$ for $i = 1,2,\ldots,n$.

Accordingly, Verify in the Data Access Phase is modified as below:

Verify: the data requester checks the freshness of ${t_i}^2$, recovers $h_{3i} = H_3(PID_i||PK_i||P_{pub})$ and $h_{4i} = H_4(c_{po_i}||PID_i||R_i||PK_i||{t_i}^2)$. Then the requester verifies whether $\delta_i P = A_i + P_{pub} + h_{3i}X_i + h_{4i}R_i$. The data requester can also check a set of messages simultaneously as in AggregateVerify to improve verification efficiency.

Security Analysis to PAASH$^+$

In the above improved CLAS scheme, a Type I adversary $\mathcal{A}_1$ can forge a valid signature in two ways, i.e., by computing the partial secret key $PSK_i = (A_i, \theta_i)$ generated by the RA or bypassing it. Although $\mathcal{A}_1$ can obtain the public key corresponding to PSK_i, the computation of the partial secret key θ_i can be viewed as solving ECDLP. The second method is to bypass θ_i. Namely, $\mathcal{A}_1$ can use some algebraic relation that may exist in the verification equation (i.e., $\delta_i P = A_i + P_{pub} + h_{3i}X_i + h_{4i}R_i$) to eliminate P_{pub}. To do this, $\mathcal{A}_1$ can only change the value of X_i (or R_i) because A_i and P_{pub} are generated by the RA. Note that $h_{3i} = H_3(PID_i||PK_i||P_{pub})$ and $h_{4i} = H_4(c_{po_i}||PID_i||R_i||PK_i||{t_i}^2)$. X_i and h_{3i} (or R_i and h_{4i}) are bound together, any modification to X_i (or R_i) will make the equation invalid.

A Type II adversary $\mathcal{A}_2$ also has two methods to forge a signature. The first method is to obtain the secret value x_i generated by the target user P_i. Although $\mathcal{A}_2$ can obtain the partial public key X_i corresponding to x_i, the computation of x_i can also be stated as solving ECDLP. The second approach is to bypass x_i. That is, $\mathcal{A}_2$ may use some algebraic relation in the above verification equation to eliminate X_i. To do this, it needs to change A_i, R_i, or P_{pub}. However, A_i has been made public as part of the user's public key; R_i and h_{4i} as well as P_{pub} and $h_{3i}X_i$ are bound together, respectively. Therefore, $\mathcal{A}_2$ cannot execute the modification.

By eliminating the algebraic relation between the public parameters in the verification equation, the suggested scheme can be proved secure through the theorem in Ogundoyin et al.'s security model:

Theorem 1. *If the ECDLP problem is hard, then the improved CLAS is unforgeable against Type I and Type II adversaries as defined by Ogundoyin et al. in [2] in the random oracle model.*

However, we omit the concrete proof here to avoid duplication of work.

By adopting the improved CLAS scheme, a secure PAASH$^+$ scheme is achieved. Table 2 makes a simple comparison between SSH in [13], PAASH in [2], and the suggested PAASH$^+$ in terms of features.

Table 2. Comparison of features with several related schemes.

Scheme	Correctness	Resist Type I Attack	Resist Type II Attack	Authentication	Access Control	Confidentiality
[13]	✓	✓	×	×	✓	✓
[2]	×	×	✓	×	✓	✓
PAASH$^+$	✓	✓	✓	✓	✓	✓

4. Cryptanalysis of the ECACS Scheme in [19]

4.1. Review of the ECACS Scheme

In [19], Benil et al. proposed a public verification and auditing scheme called ECACS to ensure EHRs security. Their design includes sixteen algorithms: Setup, Partial-Private-Key-Generation, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Sign, Verify, Aggregate, Aggregate-Verify, Store, Audit, Challenge, Proof-Generation, Proof-Verify, Log-Generation, and Check-Log-and-Verify. The first nine algorithms constitute a certificateless public verification (CPV) scheme. The remaining algorithms form a certificateless public auditing (CPA) scheme, in which the cloud server is allowed to generate proof of possession of data, and an auditor can check the correctness of the proof. The CPA scheme is built on top of the CPV scheme. In this section, we simply review the CPV scheme.

Five entities are involved in their construction:

- Key generation center (KGC): the KGC is the trusted entity that generates the system parameters and partial public/private key pairs to all other four entities according to their registration.
- Data owner: a data owner represents the patient who collects his/her healthcare data and uploads the EHR data to the cloud server for storage.
- User: a user could be a doctor or a medical researcher who wants to query the patient's EHR data.
- Medical cloud server (MCS): the MCS is managed by the cloud service provider. It stores a large amount of EHR data sent by the users.
- Third-party auditor (TPA): assigned by the data owner, a TPA is responsible for auditing the integrity of the stored EHR data. This is achieved by sending the audit challenge message to the server, and then the server replays back with a proof message.

We now describe Benil et al.'s CPV scheme, which consists of nine algorithms. Note that, in essence, the CPV scheme is a CLAS scheme. We use the same symbols as in their construction:

- Setup:
  1. Choose a composite-order bilinear group [24] and a mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Let q and P be the order and a generator of $\mathbb{G}$, respectively.
  2. Select $x \in Z_q^*$ as the system master private key and compute the corresponding master public key $P_{Pub} = xP$.
  3. Let $H_1 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$, and $H_3 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$ be three cryptographic hash functions.

4. Keep x as private and publish the system parameters as $S_{para} = \{e, q, P, P_{Pub}, H_1, H_2, H_3\}$.

- Partial-Private-Key Generation:
  1. Server partial private-key generation: it takes S_{para}, the identity of cloud server ids and x as input. The KGC randomly selects $r_{ids} \in Z_q^*$ and computes $R_{ids} = r_{ids}P$, $h_{ids} = H_1(R_{ids}||ids||P_{ids})$, and $d_{ids} = r_{ids} + h_{ids}x$. It then outputs and sends d_{ids}, R_{ids} to the cloud server.
  2. Data owner partial private-key generation: it takes S_{para}, the identity of data owner ido, and x as input. The KGC randomly selects $r_{ido} \in Z_q^*$ and computes $R_{ido} = r_{ido}P$, $h_{ido} = H_1(R_{ido}||ido||P_{ido})$, and $d_{ido} = r_{ido} + h_{ido}x$. It then outputs and sends d_{ido}, R_{ido} to the data owner.
  3. User partial private-key generation: it takes S_{para}, the identity of user idu and x as input. The KGC randomly selects $r_{idu} \in Z_q^*$ and computes $R_{idu} = r_{idu}P$, $h_{idu} = H_1(R_{idu}||idu||P_{idu})$, and $d_{idu} = r_{idu} + h_{idu}x$. It then outputs and sends d_{idu}, R_{idu} to the user.
- Set-Secret-Value:
  1. Set the secret value of the server: the server randomly chooses its secret value $Y_{ids} \in Z_q^*$.
  2. Set the secret value of the data owner: the data owner randomly chooses its secret value $Y_{ido} \in Z_q^*$.
  3. Set the secret value of the user: the user randomly chooses its secret value $Y_{idu} \in Z_q^*$.
- Set-Private-Key:
  1. Set the private key of the server: the server takes S_{para}, d_{ids}, and Y_{ids} as input. Then its private key is $dk_{ids} = (ids, x_{ids})$.
  2. Set the private key of the data owner: the data owner takes S_{para}, d_{ido}, and Y_{ido} as input. Then its private key is $dk_{ido} = (ids, x_{ido})$.
  3. Set the private key of the user: the user takes S_{para}, d_{idu}, and Y_{idu} as input. Then its private key is $dk_{idu} = (idu, x_{idu})$.
- Set-Public-Key:
  1. Set the public key of the server: taking as input S_{para} and Y_{ids}, the server computes $P_{ids} = Y_{ids}P$. The public key of the server is $PK_{ids} = (P_{ids}, Y_{ids})$.
  2. Set the public key of the data owner: taking as input S_{para} and Y_{ido}, the data owner computes $P_{ido} = Y_{ido}P$. The public key of the server is $PK_{ido} = (P_{ido}, Y_{ido})$.
  3. Set the public key of the user: taking as input S_{para} and Y_{idu}, the data owner computes $P_{idu} = Y_{idu}P$. The public key of the server is $PK_{idu} = (P_{idu}, Y_{idu})$.
- Sign: the user with identity idu takes S_{para}, Y_{idu}, and the state information Δ, and dK_{idu} as input. To sign a message M, it performs the following:
  1. Choose $r_i \in Z_q^*$ and compute $R_{id} = r_{id}P$.
  2. Compute $B = H_2(\Delta)$ and $h_i = H_3(idu_i||M_i||PK_{idu,i}||R_i)$.
  3. Compute $V_i = d_{idu} + r_iBM + h_iY_{idu}P_{Pub}$.
  4. Output the signature $\sigma = (R_i, V_i)$.

  Then the user uploads the message-signature pair to the cloud server. The server then verifies its validity via Verify.
- Verify: it takes as input S_{para}, $\sigma = (R_i, V_i)$, and $(M_i, \Delta, PK_{id}, T)$ as input. The server first computes (h_{idu}, B, h_i) as the same as in Sign. Then it check whether $e(V_i, P) = e(h_{idu}R_{idu} + h_iP_{Pub}P_{idu})e(r_iP, BM_i)$ holds. It accepts the signature if the verification equation holds and rejects otherwise.
- Aggregate: it takes as input S_{para}, Δ, and n distinct signatures σ_i, $i = 1, \ldots, n$ on different messages M_i from different users with corresponding identities idu_i and public

keys $PK_{idu,i}$. The aggregator computes $V = \sum_{i=1}^{n} V_i$ and outputs $(\{Y_{id}, M_i, P_{idu}, R_i\}_{i=1}^{n}, \Delta, V)$ as the aggregate signature for n messages.

- Aggregate-Verify: taking as input S_{para}, $(\{Y_{id}, M_i, P_{idu}, R_i\}_{i=1}^{n}, \Delta, V)$, and tuples $(idu_i, PK_{idu\,i})$ for $i = 1, 2, \ldots, n$, it operates as follows:
  1. Compute $h_{idu} = H_1(R_{idu}||idu||P_{idu})$, $B = H_2(\Delta)$, and $h_i = H_3(idu_i||M_i||PK_{idu,i}||R_i)$.
  2. Check whether $e(V, P) = e(\sum_{i=1}^{n}(h_{idu}R_{idu} + h_i P_{Pub} P_{idu})) \cdot e(\sum_{i=1}^{n}(r_i PBM_i))$ holds.
  3. Return true if the above equation holds and return false otherwise.

4.2. Scheme Analysis

We show some drawbacks of Benil et al.'s CPV scheme, which will break its correctness.

1. According to the role definition, Sign should be executed by the data owner (i.e., the patient) rather than the user (e.g., the doctor).
2. *The calculation of partial private-key generation for the server, the data owner, and the user may be wrong.* In Partial-Private-Key-Generation (i.e., the second algorithm in the scheme), to generate a partial private key for the server, the KGC randomly selects $r_{ids} \in Z_q^*$ and computes $R_{ids} = r_{ids}P$, $h_{ids} = H_1(R_{ids}||ids||P_{ids})$. It then computes and sets $d_{ids} = r_{ids} + h_{ids}x$ as the server's partial private key. In this process, the KGC needs to use P_{ids} for obtaining h_{ids}; however, the value P_{ids} is not defined before. The same problem occurs when generating the partial private keys for the data owner and the user.
3. *The setting when generating private keys for the server, the data owner, and the user may be wrong.* To generate its private key, the server takes the system parameters S_{para}, the partial private-key d_{ids}, and the secret value Y_{ids} as input and sets its private key as $dk_{ids} = (ids, x_{ids})$. However, x_{ids} is not defined before. Moreover, the partial private-key d_{ids} does not seem to be used for constructing the full private key. The same problem occurs when setting the partial private keys for the data owner and the user.
4. *The setting when generating public keys for the server, the data owner, and the user may be wrong.* To set the public key of the server, the algorithm Set-Public-Key takes as input the system public parameters S_{para} and a secret value Y_{ids} chosen by the server in Set-Secret-Value. It computes $P_{ids} = Y_{ids}P$, and the public key of the server is $PK_{ids} = (P_{ids}, Y_{ids})$. In this setting, the secret value is a part of the public key, which may affect the security of the scheme.
5. *There are some undefined definitions in the signing process.* In Sign, a piece of state information Δ and a value T are bound to the signature. One may infer that T is the timestamp; however, one cannot guess what state information Δ is used to store. Additionally, the signer needs to compute $R_{id} = r_{id}P$. However, neither R_{id} nor r_{id} has been defined. According to the description of Sign and Verify, the equation should be $R_i = r_i P$.
6. *The verification equation in* Verify *does not hold.* In the CPV scheme, a single signature for a message is valid if the verification equation $e(V_i, P) = e(h_{idu}R_{idu} + h_i P_{Pub} P_{idu})e(r_i P, BM_i)$ holds. The authors in [19] claimed its correctness as follows:

$$\begin{aligned}
e(V_i, P) &= e(d_{idu} + r_i BM_i + h_i Y_{idu} P_{Pub}, P) \\
&= e(d_{idu}, P) \cdot e(r_i BM_i, P) \cdot e(h_i Y_{idu} P_{Pub}, P) \\
&= e(r_{idu} + h_{idu}x, P) \cdot e(r_i BM_i, P) \cdot e(h_i Y_{id} P_{Pub}, P) \\
&= e(h_{idu}x, r_{idu}P) \cdot e(xP) \cdot e(r_i P, BM_i) \cdot e(h_i Y_{id} P, P_{Pub}) \\
&= e(h_{idu}, R_{idu}) \cdot e(P_{Pub}) \cdot e(r_i P, BM_i) \cdot e(h_i P_{idu}, P_{Pub}) \\
&= e(h_{idu}R_{idu}P_{Pub}) \cdot e(r_i P, BM_i) \cdot e(h_i P_{Pub}, P_{idu}) \\
&= e(h_{idu}R_{idu} + h_i P_{Pub} P_{idu}) e(r_i P, BM_i).
\end{aligned}$$

However, the equation does not hold from the fourth line according to the computation rule of bilinear pairing [25], which disproves their claim. The same error exists in Aggregate-Verify.

Obviously, the above analysis shows the proposed CPV scheme is incorrect. It is also for this reason that we do not offer any suggestions for the scheme's improvement. Moreover, since the CPA scheme is built on top of the CPV scheme, its correctness also cannot be guaranteed.

Real-world dangers. Our analysis shows that the proposed protocol fails to achieve correctness. This means that it cannot achieve the security and privacy properties claimed by Benil et al. If the scheme in [19] is adopted in reality, the patient's privacy-sensitive EHR data will be exposed to unprotected environments. For example, because the CPV scheme loses its function, the authenticity and integrity of the shared EHR data cannot be assured. In this situation, malicious entities may tamper with the real data. Moreover, this may mislead doctors to make an incorrect diagnosis, which is extremely dangerous for the patient's life.

5. Conclusions

In this work, we analyzed two recent privacy-preserving authentication schemes for smart healthcare applications, i.e., Ogundoyin et al.'s PAASH in [2] and Benil et al.'s ECACS scheme in [19]. More concretely, we observed that the underlying CLAS scheme in PAASH construction cannot achieve correctness and unforgeability. We found that the batch verification process in their CLAS scheme was incorrect because the verification equation cannot be validated. In addition, our attack showed that any Type I attacker can impersonate a patient to forge a valid signature on his false healthcare information, which may lead healthcare professionals to make incorrect diagnoses of a patient's health condition. We discussed the reason for our attack and suggested an improved PAASH$^+$ scheme, which overcomes the design flaws of PAASH and simultaneously achieves aggregate authentication, fine-grained access control, and data confidentiality. In addition, we demonstrated that Benil et al.'s ECACS scheme had many drawbacks and was incorrect. Our analysis showed that both the above two recent schemes are not suitable to be deployed in practical smart health applications. We hope our analysis will improve the design of such schemes in future work.

Author Contributions: Conceptualization, F.X.; Methodology, F.X.; Writing—original draft, F.X.; Writing—review and editing, J.L. and R.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No data was used for the research described in the article.

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

References

1. Estopace, E. IDC Forecasts Connected IoT Devices to Generate 79.4 ZB of Data in 2025. 2019. Available online: https://futureiot.tech/idc-forecasts-connected-iot-devices-to-generate-79-4zb-of-data-in-2025/ (accessed on 19 June 2023).
2. Ogundoyin, S.O.; Kamil, I.A. PAASH: A privacy-preserving authentication and fine-grained access control of outsourced data for secure smart health in smart cities. *J. Parallel Distrib. Comput.* **2021**, *155*, 101–119. [CrossRef]
3. Zhu, F.; Yi, X.; Abuadbba, A.; Khalil, I.; Nepal, S.; Huang, X. Authenticated Data Sharing With Privacy Protection and Batch Verification for Healthcare IoT. *IEEE Trans. Sustain. Comput.* **2023**, *8*, 32–42. [CrossRef]
4. Boneh, D.; Gentry, C.; Lynn, B.; Shacham, H. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In Proceedings of the EUROCRYPT 2003, Warsaw, Poland, 4–8 May 2003; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2656, pp. 416–432.
5. Shen, L.; Ma, J.; Liu, X.; Wei, F.; Miao, M. A Secure and Efficient ID-Based Aggregate Signature Scheme for Wireless Sensor Networks. *IEEE Internet Things J.* **2017**, *4*, 546–554. [CrossRef]
6. Al-Riyami, S.S.; Paterson, K.G. Certificateless Public Key Cryptography. In Proceedings of the ASIACRYPT 2003, Taipei, Taiwan, 30 November–4 December 2003; Lecture Notes in Computer Science; Laih, C., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2894, pp. 452–473.

7. Gayathri, N.B.; Gowri, T.; Kumar, P.R.; Rahman, M.Z.U.; Reddy, P.V.; Lay-Ekuakille, A. Efficient and Secure Pairing-Free Certificateless Aggregate Signature Scheme for Healthcare Wireless Medical Sensor Networks. *IEEE Internet Things J.* **2019**, *6*, 9064–9075. [CrossRef]
8. Liu, J.; Wang, L.; Yu, Y. Improved Security of a Pairing-Free Certificateless Aggregate Signature in Healthcare Wireless Medical Sensor Networks. *IEEE Internet Things J.* **2020**, *7*, 5256–5266. [CrossRef]
9. Zhan, Y.; Wang, B.; Lu, R. Cryptanalysis and Improvement of a Pairing-Free Certificateless Aggregate Signature in Healthcare Wireless Medical Sensor Networks. *IEEE Internet Things J.* **2021**, *8*, 5973–5984. [CrossRef]
10. Yang, W.; Wang, S.; Mu, Y. An Enhanced Certificateless Aggregate Signature Without Pairings for E-Healthcare System. *IEEE Internet Things J.* **2021**, *8*, 5000–5008. [CrossRef]
11. Qiao, Z.; Yang, Q.; Zhou, Y.; Yang, B.; Zhang, M. A Novel Construction Of Certificateless Aggregate Signature Scheme For Healthcare Wireless Medical Sensor Networks. *Comput. J.* **2022**. [CrossRef]
12. Yang, X.; Wen, H.; Diao, R.; Du, X.; Wang, C. Improved Security of a Pairing-Free Certificateless Aggregate Signature in Healthcare Wireless Medical Sensor Networks. *IEEE Internet Things J.* **2023**, *10*, 10881–10892. [CrossRef]
13. Zhang, Y.; Deng, R.H.; Han, G.; Zheng, D. Secure smart health with privacy-aware aggregate authentication and access control in Internet of Things. *J. Netw. Comput. Appl.* **2018**, *123*, 89–100. [CrossRef]
14. Chen, X.; He, D.; Khan, M.K.; Luo, M.; Peng, C. A Secure Certificateless Signcryption Scheme Without Pairing for Internet of Medical Things. *IEEE Internet Things J.* **2023**, *10*, 9136–9147. [CrossRef]
15. Ren, R.; Su, J. A Security-Enhanced and Privacy-Preserving Certificateless Aggregate Signcryption Scheme-Based Artificial Neural Network in Wireless Medical Sensor Network. *IEEE Sens. J.* **2023**, *23*, 7440–7450. [CrossRef]
16. Zhu, F.; Yi, X.; Abuadbba, S.; Khalil, I.; Yang, X.; Nepal, S.; Huang, X. Privacy-Preserving Authentication for Tree-Structured Data with Designated Verification in Outsourced Environments. In Proceedings of the ProvSec 2020, Singapore, 29 November–1 December 2020; Springer: Berlin/Heidelberg, Germany, 2020; Volume 12505, pp. 145–165.
17. Domadiya, N.; Rao, U.P. Improving healthcare services using source anonymous scheme with privacy preserving distributed healthcare data collection and mining. *Computing* **2021**, *103*, 155–177. [CrossRef]
18. Guo, H.; Li, W.; Nejad, M.M.; Shen, C. A Hybrid Blockchain-Edge Architecture for Electronic Health Record Management with Attribute-Based Cryptographic Mechanisms. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 1759–1774. [CrossRef]
19. Benil, T.; Jasper, J. Cloud based security on outsourcing using blockchain in E-health systems. *Comput. Netw.* **2020**, *178*, 107344. [CrossRef]
20. Galbraith, S.D.; Gaudry, P. Recent progress on the elliptic curve discrete logarithm problem. *Des. Codes Cryptogr.* **2016**, *78*, 51–72. [CrossRef]
21. Gafif, H.E.; Meddah, N.; Toumanari, A. A Lightweight Ciphertext-Policy Attribute-Based Encryption for Fine-Grained Access Control. In *Proceedings of the International Conference on Advanced Intelligent Systems for Sustainable Development*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 13–23.
22. Shim, K. Security models for certificateless signature schemes revisited. *Inf. Sci.* **2015**, *296*, 315–321. [CrossRef]
23. Wu, G.; Zhang, F.; Shen, L.; Guo, F.; Susilo, W. Certificateless aggregate signature scheme secure against fully chosen-key attacks. *Inf. Sci.* **2020**, *514*, 288–301. [CrossRef]
24. Lewko, A.B. Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting. In Proceedings of the Advances in Cryptology-EUROCRYPT 2012, Cambridge, UK, 15–19 April 2012; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7237, pp. 318–335.
25. Paterson, K.G. Cryptography from pairings. *Lond. Math. Soc. Lect. Note Ser.* **2006**, *317*, 215.