axioms

Article

Methods to Encrypt and Authenticate Digital Files in Distributed Networks and Zero-Trust Environments

Bertrand Cambou [1,*], Christopher Philabaum [1], Jeffrey Hoffstein [2,3] and Maurice Herlihy [2,3]

1 School of Informatics Computing and Cyber Systems, Northern Arizona University, Flagstaff, AZ 86011, USA
2 Department of Computer Science, Brown University, Providence, RI 02901, USA; jeffrey_hoffstein@brown.edu (J.H.); maurie_herlihy@brown.edu (M.H.)
3 Department of Mathematics Brown University, Providence, RI 02901, USA
* Correspondence: bertrand.cambou@nau.edu

Abstract: The methods proposed in this paper are leveraging Challenge–Response–Pair (CRP) mechanisms that are directly using each digital file as a source of randomness. Two use cases are considered: the protection and verification of authenticity of the information distributed in storage nodes and the protection of the files kept in terminal devices operating in contested zero-trust environments comprised of weak signals in the presence of obfuscating electromagnetic noise. With the use of nonces, the message digests of hashed digital files can be unique and unclonable; they can act as Physical Unclonable Functions (PUF)s in challenge–response mechanisms. During enrollment, randomly selected "challenges" result in unique output data known as the "responses" which enable the generation and distribution of cryptographic keys. During verification cycles, the CRP mechanisms are repeated for proof of authenticity and deciphering. One of the main contributions of the paper is the development of mechanisms accommodating the injection of obfuscating noises to mitigate several vectors of attacks, disturbing the side channel analysis of the terminal devices. The method can distribute error-free cryptographic keys in noisy networks with light computing elements without relying on heavy Error Correcting Codes (ECC), fuzzy extractors, or data helpers.

Keywords: obfuscation; validation; authentication; digital file; cryptography; electronic noise; challenge; response

MSC: 94A60; 94A62; 94B70; 94C12

check for updates

Citation: Cambou, B.; Philabaum, C.; Hoffstein, J.; Herlihy, M. Methods to Encrypt and Authenticate Digital Files in Distributed Networks and Zero-Trust Environments. *Axioms* **2023**, *12*, 531. https://doi.org/10.3390/axioms12060531

Academic Editor: Sidney A. Morris

Received: 15 April 2023
Revised: 9 May 2023
Accepted: 25 May 2023
Published: 29 May 2023

1. Introduction and Background Information

The objective of this work is to develop novel mechanisms that offer proof of authenticity in distributed networks and protect terminal devices operating in zero-trust areas. In our differentiated approach, we consider several complementary remedies: (i) one-time use keys for each transaction, (ii) generating keys from the message digest of each file, and (iii) in zero-trust networks; transmit the data feeding a CRP mechanism through noisy wireless channels to enable the decryption on-demand of digital files. All proposed remedies are designed to handle the erratic bits without ECC. In this section, we are summarizing some of the relevant work conducted by prior authors to secure digital files and use CRP mechanisms. Storing and distributing the cryptographic keys needed to protect sensitive information and terminal devices introduces a level of risk. Of concern are replays, man in the middle attacks, loss of information in the network, side channel analysis and physical loss to the opponent of a terminal device [1–3]. Storing non-encrypted files in the terminal introduces the same risk level as storing the secret keys that decrypt the cipher texts of these files. In distributed networks, the clients usually store the public–private key pairs in their terminal devices, which presents an element of risk. The opponents also inject noise to disturb the wireless communication between the ground operation and terminal device, introducing difficulty in the distribution of cryptographic keys without heavy ECC, fuzzy extractors, or data helpers, all of which are leaking information [4–6].

Blockchain technology with digital signatures offers protection in exposed networks, such as the non-alterability and traceability of chains of transactions [7]. Digital Signature Algorithms (DSAs) further enhance security by associating the blockchains to a set of known users and their public keys [8]. Blockchains with digital signatures are exposed to a variety of possible attacks that introduce risk in validating authenticity, specifically when operating outside a reliable network [9,10]. Concerns include possible difficulties in verifying the legitimacy of a file, the theft of private keys, and blockchains that may be forged by associating altered messages with an altered message digest. To prove the authenticity of a message in zero-trust networks, it is important to add layers of security without imposing excessive computing power consumption.

Background information in the area of file verification and distributed networks is presented in refs. [11–21]. In [11], a Challenge Response Authentication (CRA) of physical layers is described to avoid exposing passwords. The work presented in [12] shows the methods to provide verification of information in a ledger. The authors of [12] define a “cryptographic challenge nonce” and a protocol that enables third parties to verify information; the technology is based on digital signatures with public/private keys. In [13], document tracking schemes on a distributed ledger [13] are presented and details how to insert unique identifiers and hash values for this tracking; a processor coupled with the storage device is used in the protocols. In [14,15], the authors present a secure exchange of signed records. These records contain digital signatures of the senders and their public keys, which enhances security. Methods to authenticate data based on proof verification are shown in [16–18]. Some of the data may be external to the network, protected by a blockchain and DSA with public keys. Uhr et al. describe the methods to verify certificates based on blockchains. The system compares the content of a previously known certificate by a financial institution with the content of a certificate obtained at the point of use. Sheng et al. describe computationally efficient methods to audit transactions. The methods are based on blockchains, Bloom filters, and digital signatures. Information such as using a timestamp as an entry and wallet address are part of the scheme. In [19], a method to validate documents with blockchains and other information is presented. Only portions of the documents are shared; the full document is not revealed. Publication [20] describes contract agreement verifications. Electronic signatures are inserted in the contracts and the blockchains with a method to share public/private key pairs. Publication [21] presents a management system protecting enterprise data with blockchains. The computing device selects what data to incorporate in the blockchain to offer adequate protections.

The CRPs developed for PUFs provide relevant background information for our work even if they do not use a digital file [22–30]. Authentication based on a challenge, a response, a PUF, and machine learning are described in [22]. The development of secure digital signatures using PUF devices with reduced error rates is presented in [23,24]. Cryptographic protocols with PUF-based CRPs are shown in [25,26]. Biometry with a CRP mechanism is suggested in [27,28]. CRP mechanisms and CRA schemes are also applied to protect centralized or distributed networks [29–31]. An example of a scheme is the formation of a look-up table with indexes in the left column and passwords in the right column. The challenges point at an index to be paired with an associated password, which avoids the problem of having to change the passwords periodically.

Some of the contributions of this work include a novel approach based on CRP schemes to protect and authenticate each digital file individually, with the objective to mitigate several vectors of attack. The idea to inject obfuscating noise during ground-to-terminal communication is developed to limit the ability of opponents to share the same wireless network for side channel analysis. To facilitate a quick adoption of the novel methods, all encryption algorithms are standardized cryptographic algorithms recommended by the National Institute of Standards and Technologies (NIST), as well as the codes under consideration for standardization by NIST for post-quantum cryptography (PQC). The paper is organized as follows:

- [Section 2] The first layer of technology needed for our proposed schemes is described, namely the design of the CRP mechanisms directly based on digital files. The algorithms extracting the responses from randomly selected challenges are presented as well as methods to generate cryptographic keys from the CRPs.
- [Section 3] The protocols to verify the authenticity of digital files with CRP mechanisms in distributed networks are presented. The algorithms for enrolment and for verification are detailed. An example of a use case that is using these protocols with an agent, storage node, and smart contract is given. A security analysis, in which we list potential issues and remedies, is also provided.
- [Section 4] The protocols to securely distribute digital files to terminal devices exposed to zero-trust networks are suggested. A use case that distributes cryptographic keys while injecting erratic bits is presented.
- [Section 5] The potential problems created by residual erratic bits in the recovery keys are examined in detail. A model is developed and verified, and light error management schemes are suggested.

Finally, Section 6 presents the summary and future work. A table listing the acronyms used herein is presented in Attachment A, followed by the list of references.

2. CRP Mechanism Based on Digital Files

Methods to design digital file-based CRP mechanisms will now be presented. The method detailed in Section 2.1 describes a generic response generation scheme for applications in distributed networks. The implementation detailed in Section 2.2 is an example of protocol offering higher levels of security in distributed networks.

2.1. Response Generation with File-Based CRP Mechanism

The input data of the CRP mechanism is derived from file F. Its ciphertext C is concatenated with nonce ω to generate a file C* of constant length $d = 2^D$, where D is the number of digits (for example, if a desired $d = 1$ million, then $D = 20$). The resulting d bits are located at addresses varying from 1 to d. Changing a single bit in file F results in a totally different stream C*. This can be performed with a variety of methods; one possible implementation is the following:

- Encrypt F with private key Sk, and a PQC scheme (PQC and DSA) under standardization by NIST [32] such as the Cryptographic Suite for Algebraic Lattices (CRYSTALS)-Kyber [33,34], CRYSTALS-Kyber [35,36], NTRU [37–39], Classic McElice [40], SPHINCS [41], and Falcon [42].
- The ciphertext is hashed with Standard Hash Algorithm (SHA)-512.
- The resulting steam is XORed with 512-bit long nonce ω.
- The function SHA algorithm and Keccack (SHAKE) is used to extend the 512-bit long stream to d-bits. The combination of SHA-3 and SHAKE is NIST-compliant [43–45].

The CRP mechanism is based on the d-bit long stream C* to generate N responses from N challenges:

- *Challenges*: A "challenge" is defined as the digital information needed to point at a particular position in the d-bit long stream C*. A stream of bits S* is generated by hashing and extending with eXtended output Function (XoF), creating a randomly selected seed S. The stream S* is segmented into N challenges $\{q_1, \ldots, q_i, \ldots, q_N\}$ that are D-bit long. The D bits of each challenges $\mathrm{q_i}$ are converted into number x_i, with $x_i \in \{1, d\}$, which is an address in C*, resulting in N addresses $\{x_1, \ldots, x_i, \ldots, x_N\}$
- *Responses:* The N addresses generate the P-bit long responses $\{r_1, \ldots, r_i, \ldots, r_N\}$. From each address x_i, P-bit long responses are generated from C*. The iterative method to find the P positions $\{x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,P}\}$, and read the P-bits is the following:

The first position is: $x_{(i,j=1)} = x_i$. The other positions $x_{(i,j)}$ are given by the linear congruent random number generator, $i \in \{1, N\}$, $j \in \{1, P\}$, see Equation (1):

$$x_{(i,j)} = (\alpha x_{(i,j-1)} + \beta) mod(2^D);\ \text{with } \alpha \text{ and } \beta \text{ prime numbers} \quad (1)$$

Algorithm 1 and the block diagram shown in Figure 1 summarize the above protocol. The output after response generation is {C, ω, S} and $\{r_1, \ldots, r_i, \ldots, r_N\}$. Such a CRP mechanism is highly secure as three totally independent streams are requested to uncover the responses which are C, ω, and S. Protecting just one of the three provides acceptable security. The computing power, or "gas price", required to run the CRP mechanism is low.

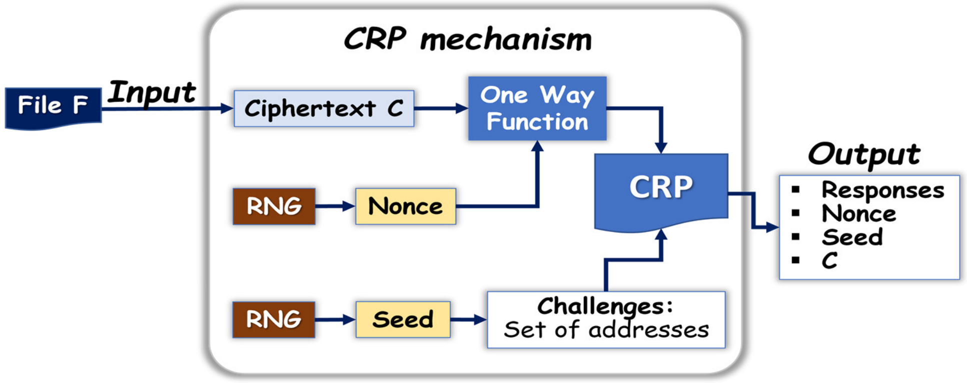

Figure 1. Block diagram of a CRP mechanism generating a set of responses. The file F is encrypted and concatenated with a nonce to compute a crypto-file. A seed generates a set of challenges.

Algorithm 1: Generate a set of responses with C

```
1:   Variable input data: file {$C$}
2:   Nonce {ω}, and stream {$S$} ← random number generator
3:   → Module 1: Generate a set of responses with C, and {ω,S}:
     3.1:  Static input data: positive integers $d, D, N, P, \alpha, \beta, d = 2^D$ and $\alpha, \beta$ are prime
     3.2:  $MD$ ←Hash (C) (ex: SHA-256)
     3.3:  C* ← XOV(concatenate ($MD$, ω)) (ex: SHAKE)
           [Comment: Organize C* with bits located at addresses 1 to d]
     3.4:  S*← XOV ($S$); with S* is a (N × $D$)-bit long stream
     3.5:  $\{q_1, \ldots, q_i, \ldots, q_N\}$ ← S*; Split S* into N, D-bit long, challenges $q_i$; i ∈ {1, N})
     3.6:  $\{x_1, \ldots, x_i, \ldots, x_N\} \leftarrow \{q_1, \ldots, q_i, \ldots, q_N\}$; for $N$ positions $x_i$ in C*; $x_i \in \{1, d = 2^D\}$
           $\{r_1, \ldots, r_i, \ldots, r_N\} \leftarrow \{x_1, \ldots, x_i, \ldots, x_N\}$; for all $N$, $P$-bit long, responses $r_i$
     3.7:  For each position $x_i$ generate P-bit long response $r_i$in the following way:
           • $\left\{x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,P}\right\} \leftarrow x_i$; j ∈ {1, P}
           [comment: find P positions $x_{i,j}$ in C* with congruent linear generator]
               ○ if j = 1, then $x_{(i,j=1)} = x_i$
               ○ Else, iterate: $x_{(i,j)} = \left(\alpha x_{(i,j-1)} + \beta\right) mod\left(2^D\right)$
           • $r_i \leftarrow \left\{x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,P}\right\}$;
           [Comment: read the P positions $x_{i,j}$ in C* to generate P-bit long response $r_i$]
4:   Output: $C$, {$S$, ω,} and the $N$ responses $\{r_1, \ldots, r_N\}$
```

The probability to point several times to the same position in C* is non-negligeable. This exposes the protocol to a frequency analysis and creates collisions, as several responses r_i are then identical. One solution tracks the positions with an index to generate different responses when several challenges are pointing at the same position. For example, when a position is selected a second time, the β of Algorithm 1 step 3.7 can be replaced by $\beta + 1$ which is a sufficient offset to generate a distinct response; iterating, the β is replaced by $\beta + 2$ during the third collision, et cetera.

2.2. Generation of an Orderly Subset of Responses

In the scheme described in Section 2.1, the responses generated from the CRP mechanism are directly converted into cryptographic keys K to encrypt a message M. To enhance security, we developed a scheme in which the ephemeral key K is picked randomly, independent from the CRP mechanism. Algorithm 2 for encryption and Algorithm 3 for decryption are summarized in Figure 2, and shown as follows:

Algorithm 2: Generate a subset of responses with C, and encrypt M

```
1:   Variable input data: $\{C\}$, $\{M\}$
2:   Nonce $\{\omega\}$, and stream $\{S\} \leftarrow$ random number generator
3:   $\rightarrow$ Use Module 1: Generate a set of responses with C, and $\{\omega, S\}$:
             Output data: All $N$, $P$-bit long, responses $\{r_1, \ldots, r_N\}$.
4:   $\rightarrow$ Module 2: Encrypt M and generate a subset of responses with $\{r_1, \ldots, r_N\}$:
     4.1:  key $K$ with f states of "1": $\{k_1, \ldots, k_N\} \leftarrow$ random number generator
     4.2:  $M^* \leftarrow$ encrypt($M$, $K$)
     4.3:  Filter subset of f responses $\{r'_1, \ldots, r'_f\}$ located at positions of $K$ with state of "1"
     4.4:  Erase $M$, $K$, and the $N - f$ responses located at positions of K with state of "0"
5:   Output: $C$, $\{S, \omega, M^*\}$ and the f responses $\{r'_1, \ldots, r'_f\}$
```

Algorithm 3: Decrypt M with C and the subset of f responses

```
1:   Variable input data: $C$, $\{S, \omega, M^*\}$ and the $f$, $P$-bit long, responses $\{r'_1, \ldots, r'_f\}$
2:   $\rightarrow$ Use Module 1: Generate a set of responses with C, and $\{\omega, S\}$:
             Output data: the $N$, $P$-bit long, responses $\{r_1, \ldots, r_i, \ldots, r_N\}$, $i \in \{1, N\}$
3:   $\rightarrow$ Module 3: Decrypt $M$ from $M^*$ with the $f$ responses $\{r'_1, \ldots, r'_j, \ldots, r'_f\}$, $j \in \{1, P\}$:
     3.1:  Retrieve key $K$ by comparing the $N$ responses $r_i$ with the subset of f responses $r'_j$:
           •  If $r_i$ matches at least one response $r'_i$, then $k_i = 1$
           •  Else, $k_i = 0$
     3.2:  $M \leftarrow$ Decrypt($M^*$, $K$)
4:   Output: $M$
```

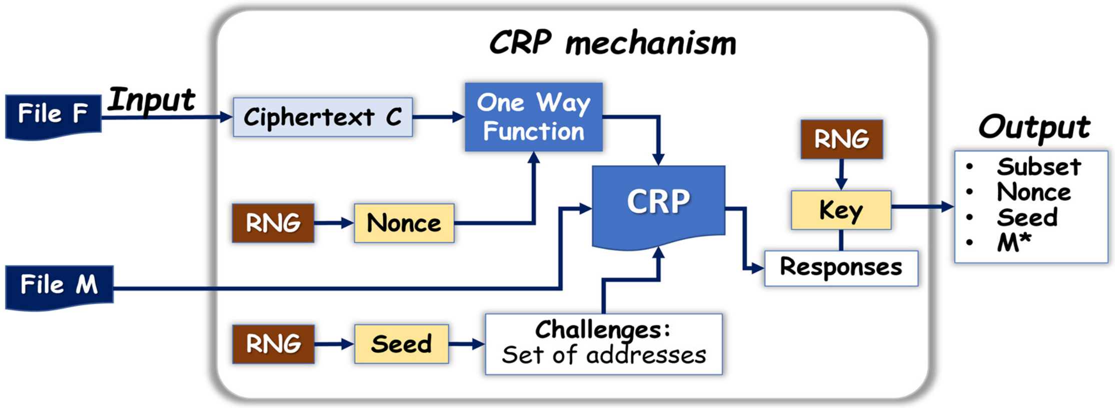

Figure 2. CRP mechanism with subset of responses. The N responses are generated with the CRP. The subset of f responses corresponds to the position of the randomly picked K with a state of 1.

The key K, which is randomly picked here, has two purposes: encrypting M and generating an orderly subset of responses. Only the f positions of K with a state of "1" are pointing at positions in the sequence of N responses that are kept, while the responses at the $N - f$ positions of K with a state of "0" are skipped. The resulting orderly subset of f responses is kept for key recovery. Four totally independent streams are requested to uncover the responses, which are C, ω, S, and the subset of responses. The size of the responses needed to run this CRP mechanism is higher than the size as presented in

Section 2.1. For example, approximately 128 responses, that are 100-bit long, are generated when K is 256-bit long. Therefore, 1.6 Kbytes are retrieved, rather than 256 bits in the simpler protocol. We notice that the orderly subset of responses does not have a fixed length. Assuming that the random number generator of the 256-bit long K has an equal probability to get a state "0" or "1", the probability $P(X = x)$ of obtaining an x-bit long subset is shown in Figure 3. The median value of the distribution is 128 and varies approximately from 100 to 160. To obfuscate this information, we add nonces at the end of the subset to keep the length to 160 responses. The key recovery scheme as presented in Section 4 is based on a search from left to right that terminates when the sequence of 256 bits is identified; additional responses are ignored.

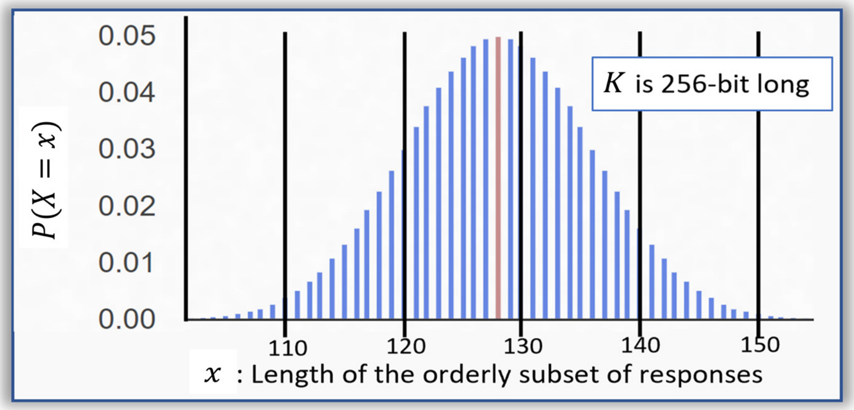

Figure 3. Probability to have the length of the subset at x when K is 256-bit long. The average length of the subsets is centered at 128 with the bulk of the distribution in the 100 to 160 range.

The injection of errors in the subset of responses is detailed in Sections 4 and 5. Provided the BERs remain below an acceptable threshold (i.e., below 25%) the subset of orderly responses can be recognized as part of the iterative search; therefore, ephemeral key K can be recovered. In zero-trust networks, such an obfuscation can disturb opponents sharing the noisy network. Certain attacks are mitigated since the noisy responses are different than the ones generated from F and the CRP mechanism.

3. Protocols Verifying the Authenticity of Digital Files in Distributed Networks

The objective of the protocols presented in this section is to allow third parties operating openly in the distributed network to validate the authenticity of a file without the over-burdening of computational resources. The suggested protocols are based on the CRP mechanism discussed in Section 2 with the same symbols. The message of authenticity M was replaced by public key Pk computed internally to add a DSA.

3.1. Description of the Protocols with CRP Mechanisms

The overall mechanism to generate the responses is described in Section 2.1. For example, if $d = 1024$, $D = 10$, $N = 32$, and $P = 8$, the responses are 256-bit long. The entropy is about 200, as the number of possible CRPs is given by Equation (2):

$$\binom{d = 1024}{N = 32} = 4.6 \times 10^{60} \approx 2^{200} \quad (2)$$

3.1.1. Enrollment Cycle

The protocol has two steps: the initial set up or enrollment performed secretly by the client or designate, and the verification of the authenticity cycle performed openly in the distributed network. Algorithm 4 for enrolling file F is shown as follows:

Algorithm 4: Enrollment cycle for file *F*

```
1:  Input data: Some file $F$
2:    Nonce $\{\omega\}$, stream $\{S\}$, and seed $\{L\} \leftarrow$ random number generator
3:    Generate ephemeral public-private key pair $\{Sk, Pk\}$ from $L$ (ex: PQC algorithm)
4:    $C \leftarrow$ Encrypt($F$, $Sk$)
5:    $M \leftarrow Pk$
6:  $\rightarrow$ Use Module 1: Generate a set of $N$ responses with $C$, and $\{\omega, S\}$:
      • Static input data: Positive integers $d$, $N$, $P$, $\alpha$, $\beta$
      • Output data: All $N$, $P$-bit long, responses $\{r_1, \ldots, r_N\}$
7:    $K \leftarrow$ concatenate$\{r_1, \ldots, r_N\}$; where $K$ is a 256-bit long key
8:  $M^* \leftarrow$ Encrypt($Pk$, $K$)
9:  Erase: $\{C^*, Sk, Pk, K\}$, and $\{r_1, \ldots, r_N\}$
10: Output: $C$, and steams $\{S, \omega, M^*\}$
```

During the enrollment of File F, a randomly selected seed L generates a key pair (Sk; Pk) with an asymmetrical algorithm such as Dilithium. The file F is encrypted with Sk to generate the ciphertext C. The randomly selected nonce ω is concatenated with C to form a d-bit long file C* that is at the center of the CRP mechanism. A randomly picked seed S is converted into a set of N, D-bit long, challenges generating a set of N, P-bit long responses. The responses are concatenated to generate the ephemeral key K, which encrypts the public key Pk and forms ciphertext M*. After completion of the enrollment cycle, the responses, C*, and keys K, Sk, and Pk are erased. The output is C and stream {S, ω, M*}.

3.1.2. Verification Cycle

The verification cycle follows the same steps as the enrollment cycle. The release of ω initiates the process, while {C, S, M*} are also needed. The replay of the CRP mechanism retrieves the responses from ω and {C, S}. The responses allow the recovery of K, the deciphering of Pk from M* with K, and the deciphering of C with Pk to retrieve F. Algorithm 5 is summarizing this protocol as shown below. A variation of this protocol utilizes a message of authenticity M, instead of Pk, to generate ciphertext M* with K; where, in this case, C is decrypted separately. F can remain secret; while the proof of authenticity of C can be public, informing peers that C is valid.

Algorithm 5: Decrypt file F

```
1:  Variable input data: $\{\omega\}$, and $\{C, S, M^*\}$
2:  $\rightarrow$ Use Module 1: Generate a set of $N$ responses with $C$, and $\{\omega, S\}$:
      • Static input data: Positive integers $d$, $N$, $P$, $\alpha$, $\beta$
      • Output data: the $N$, $P$-bit long, responses $\{r_1, \ldots, r_N\}$
3:  $K \leftarrow$ concatenate$\{r_1, \ldots, r_N\}$
4:  $Pk \leftarrow$ Decrypt($M^*$, $K$)
5:  $F \leftarrow$ Decrypt($C$, $Pk$)
6:  Output: $F$, $Pk$
```

3.2. Example of Use Case in Distributed Networks

The schemes presented above in Section 3.1 benefit clients operating in a distributed network [46–50]. In this example, suppose a client desires to allow independent parties to verify the authenticity of a transaction contained in file F. The client keeps secret F and discloses ciphertext C. The roles of the participating parties are defined as the following:

- Client: owns file F but pays the storage agent to store C for some duration [51].
- Agent: represents the client [52].
- Storage agent: paid by the client or the agent to store information [53].
- The data is public.
- Smart contract: The client's rent is kept in escrow in the contract until the rental expires. The smart contract is equipped to compute the proof of authenticity [54,55].

The client issues periodic challenges to check that the storage agent is faithful. If a challenge fails, the smart contract refunds the rent to the client. Since the smart contract's state is always public, the contract cannot keep any secret data.

3.2.1. Initial Enrollment and Distribution of the Files

The client uses a CRP mechanism during an initial set up phase, as described in Section 3.1, then distributes the information to the network. In most implementations, a compact seed replaces the challenges, as described in Section 2. The client is either equipped with the software needed to operate a CRP mechanism or employs a contract agent to perform the confidential task. From F_i and nonce ω_i, the CRP mechanism generates the ciphertext C_i and the file $M°_i$: $\{S_i, M^*_i\}$. The stream M^*_i is the ciphertext encrypted with ephemeral key K of the message of authentication M_i, which can be the public key $M_i = Pk_i$ needed for the DSA of F_i. An active participant such as an agent can take care of the distribution of C_i and $M°_i$. In Figure 4, the distribution is the following:

- The smart contract has the technical capability to perform CRP mechanisms.
- The storage node keeps C_i and $M°_i$.
- The client keeps F_i and ω_i.

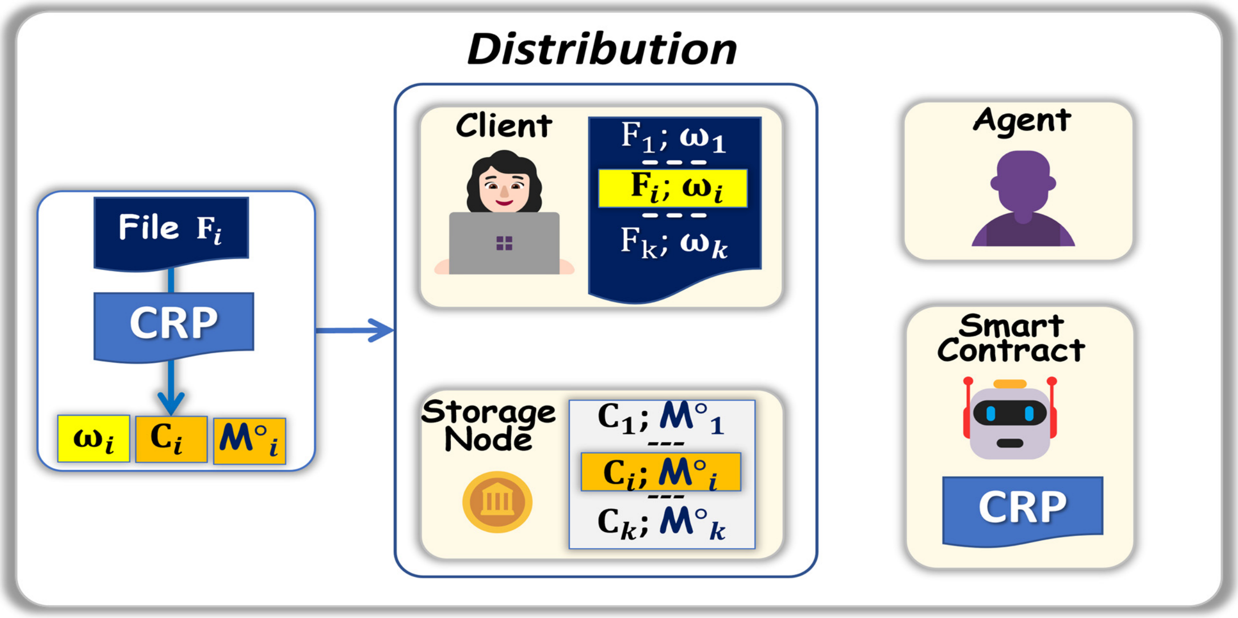

Figure 4. Block diagram of the enrollment cycle with CRP mechanism. The resulting data distributed to the agent is C_i, the cipher text of F_i, and $M°_i$: $\{S_i, M^*_i\}$, the encrypted information needed for verification of authenticity. The client keeps F_i and nonce ω_i.

3.2.2. Verification in Distributed Networks

When the verification of the authenticity of F_i is required, the client communicates ω_i to the agent. The agent then coordinates the effort between the smart contract and the storage node, executed in the protocol shown in Figure 5, which is summarized as follows:

- The smart contract, which is equipped with the CRP mechanism collects ω_i, C_i, and $M°_i = \{S_i, M^*_i\}$
- The CRP mechanism has the information needed to decrypt F_i and run the DSA verification with Pk_i.

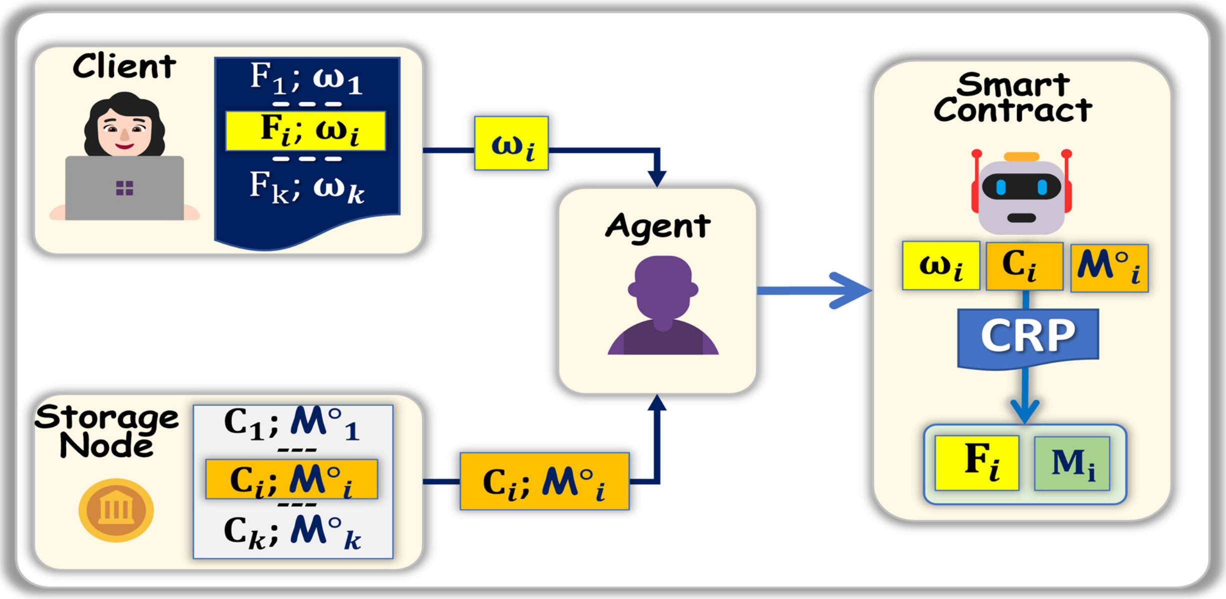

Figure 5. Block diagram of a validation cycle in a distributed network. The agent collects the information needed for the smart contract to decrypt F_i: ω_i and the stream $M°_i$: $\{S_i, M^*_i\}$. This allows the recovery of F_i, M_i to verify authenticity with DSA.

A variation of the protocol replaces Pk_i by a message of authenticity M_i such as:

"Yes, I (client X), am confirming the authenticity of ciphertext C_i".

This variation can be useful if the client desires to protect file F_i. The decryption of C_i can be performed separately, while outside the open network. A separate arrangement is then needed to share Pk_i; for example, through a public key infrastructure (PKI).

3.3. Security Analysis in Distributed Networks

The methods for verifying authenticity, as presented here, are based on the handling of several independent elements: File F, Ciphertext C, nonce ω and the stream M°. The client may restrict any of these streams while distributing the other three elements. The possibility matrix is shown in Table 1.

Table 1. Methods to organize a collaborative network.

CASE	File F	Ciphertext C	Nonce ω	Cipher Text M°: {S; M*}	What Is M?
1	Client	Storage	Client	Storage	M = Pk
2	Client	Storage	Client	Storage	M = message
3	Client	Client	Storage	Storage	M = Pk
4	Storage	Storage	Client	Storage	M = message
5	Storage	Storage	Storage	Client	M = message

The respective value of each case can be summarized as follows:

- Case 1 was discussed in Section 3.2.2 The client keeps F and ω secret after enrollment and then discloses both during verification. Such a method is valuable since the entire chain of information is obfuscated before validation. After validation, both C_1 and F_1 are public information, and the use of the DSA can further enhance the transparency of the protocol.
- Case 2 is a variation of Case 1 where M is a message of authenticity rather than a public key. The client can only disclose ω during verification, offering some homomorphic

capabilities to the scheme as the open protocols verify the authenticity of ciphertext C without openly disclosing F.

- Case 3: After enrollment, the client keeps both F and C such that the level of obfuscation is also high. During validation, the client discloses both so the smart agent can verify authenticity. This method is interesting since the client cannot lie by changing F and C after the fact.
- Case 4: After enrollment, the client only keeps ω and releases both unencrypted and encrypted files. The release of the nonce enables a full verification of authenticity to include DSA. The nonce is then used as a one-time public key.
- Case 5: This is a variation of Case 4 in which only M° is kept by the client, thus utilizing M° as a one-time public key.

Discussion: The first two cases offer both high levels of security and transparency since file F is kept secret after enrollment, while ciphertext C is public information. The client could be accused of modifying the file, if it is altered; therefore, the protocol offers non-alterability and non-repudiation. In addition to the proof of authenticity of the message, by releasing the nonce, the client also provides a form of authentication. In both cases, the storage node is keeping files that can be decrypted on-demand, acting as a repository of critical information. Examples of information are medical files, financial transactions, movies, games, and/or real estate transactions.

The second case offers some homomorphic properties since the encrypted file can be authenticated without knowing the content of the file. The decryption of the files can be performed separately and secretly in a transaction between the client and a third party. The client may, for example, secretly communicate the cryptographic key needed to retrieve the file. An example of a use case could be in the handling of medical files. The doctor keeps the nonce, and the key needed to decrypt the file, while the cipher text and encrypted message of authenticity are kept by a storage node. To allow a second doctor to access the file, the doctor openly transmits the nonce, which will confirm the authenticity of the ciphertext, then secretly transmits the key deciphering the file. That way, the doctors are not required to store the files on their server to retrieve the information as needed.

The suggested protocols are relatively well protected against attack vectors such as replays, man-in-the-middle, and side channel analysis. New nonces, key pairs Sk–Pk, and seeds are generated for each file and used only once. After a proof of authenticity cycle, the entire chain of information is openly distributed: the file, the ciphertext, nonce, seeds, public key, and message of authenticity. Even the knowledge of Sk, the private key, becomes irrelevant. This is a departure from many other DSA schemes used in cryptology in which the leak of the secret key is catastrophic.

A remaining vulnerability of the protocol presented in the first two cases is the leak of nonce ω. Adversaries are then enabled to decrypt the files, which defeats the purpose of the protocol. A potential remedy is to use Multi-Factor Authentication (MFA). The nonce can be XORed with a password before storage by the client. During the verification cycle, the client shall XOR it again with the password to retrieve the nonce. Another option is to use the additional layer of security presented below in Section 4.

4. Protocols Protecting Terminal Devices in Zero-Trust Networks

As discussed in Section 3 for distributed networks, the application of interest in this section is also based on CRP mechanisms; however, the constraints are different. The protocols cannot rely on third party validations and the information is restricted to a small group of participants. One of the objectives is to enable a controlling party to drive the deciphering of the files stored in a terminal device through a zero-trust network. The transfer of information from the controlling party to the device is obfuscated with heavy electromagnetic noise, either due to the fact that the signals are weak (unintentional) or to protect the device against a variety of attacks, and channel analysis (intentional).

4.1. Description of the Protocols with CRP Mechanisms

Starting with the CRP mechanism as described in Section 2.2 randomly pick ephemeral keys, and generate an orderly subset of the sequence of responses K_R in which obfuscating noise can be injected. The enrollment cycle should be completed in a secure environment (before entering the zero-trust network). During the validation cycles that are occurring in zero-trust networks, the terminal device can decrypt its files without ECC in order to reduce leaks of information while conserving computing power.

4.1.1. Enrollment Cycle

The input data in Algorithm 6 shown below are: d, N, P, and prime numbers α, β. The ephemeral key K is picked randomly. K is used to select the orderly sequence of f responses K_R: $\left\{r'_1, \ldots, r'_f\right\}$ from the full set of responses $\{r_1, \ldots, r_N\}$ by skipping the $N - f$ positions with a state of 0. K_R is kept secret by the controlling party. Let Kc be the stream of the nonce $\{\omega\}$, the seed S, and the ciphertext M*. Both C and Kc are distributed to the terminal device. All other information and cryptographic keys are erased, namely {C*, Sk, Pk, K} and the N-f responses with a state of 0.

Algorithm 6: Enrollment cycle for file *F*, generate subkeys *Kc* & *Kr*

```
1:   Input data: Some file $F$
2:   Nonce {ω}, stream {S}, seed {L} ← random number generator
3:   Generate ephemeral public-private key pair {$Pk$, $Sk$} from $L$ (ex: PQC algorithm)
4:   $C \leftarrow$ Encrypt($F$, $Sk$)
5:   $M \leftarrow Pk$
6:   → Use Module 1: Generate a set of $N$ responses with $C$, and {ω,$S$}:
       • Static input data: Positive integers $d$, $N$, $P$, $\alpha$, $\beta$
       • Output data: the N, P-bit long, responses $\{r_1, \ldots, r_N\}$
7:   → Use Module 2: Generate a subset of responses from $M = Pk$ and $\{r_1, \ldots, r_N\}$:
       • Key $K$ with $f$ states of "1" ← random number generator Filter the $f$ responses $\left\{r'_1, \ldots, r'_f\right\}$
       • $M^* \leftarrow$ Encrypt($M$, $K$)    Erase: {$Sk$, $Pk$, $K$}
8:   Let subkey $Kr$ be the $f$, $P$-bit long, responses $\left\{r'_1, \ldots, r'_f\right\}$
9:   Let subkey $Kc$ be the streams {$S$, ω, $M^*$}
```

4.1.2. Verification Cycle

The terminal device can verify the authenticity of C by recovering Pk with Kr and Kc, in which Kr: $\left\{r'_1, \ldots, r'_j, \ldots, r'_f\right\}$ and Kc: {S, ω, M*}.

- The first part of Algorithm 7 shown below is a replay of the enrollment cycle to generate the responses $\{r_1, \ldots, r_i, \ldots, r_N\}$ that each have a length equal to P-bit.
- The key K is recovered by comparing the full sequence of responses with the subset, as shown in module 4.

The injection of bad bits in the subset of the response, a process that can also be called "noise injection", has no effect on the outcome of the protocol provided the noise level is below the threshold T that can disturb the recovery of K.

Algorithm 7: Decrypt file *F* with subkey *Kc* and noisy subkey *Kr*

```
0:  Up to $T$ bad bits are injected in $Kr$
1:  Variable input data: C, Kc and the noisy $Kr$
2:  → Use Module 1: Generate a set of N responses with C, and {$\omega$,$S$}:
      • Static input data: Positive integers $d$, $N$, $P$, $\alpha$, $\beta$
      • Output data: the $N$, $P$-bit long, responses $\{r_1, \ldots, r_i, \ldots, r_N\}$
3:  → Module 4: Decrypt $M$ from $M$* and the $f$ noisy responses $\left\{r'_1, \ldots, r'_j, \ldots, r'_f\right\}$, $j \in$ {1, $P$}:
    3.1:  Enter both sets $\left\{r'_1, \ldots, r'_j, \ldots, r'_f\right\}$, $\{r_1, \ldots, r_i, \ldots, r_N\}$ , and $T$
    3.2:  $j = 1, i = 1$
    3.3:  While $i < N + 1$:
          Measure hamming distance $\mathrm{H}(\mathrm{r'_j}, \mathrm{r_i})$:
          • If $H(r'_j, r_i) \leq T$: $k_i = 1$; increment $i, j$ by 1
          • Else: $k_i = 0$; increment $i$ by 1
    3.4:  ($M$ = $Pk$) ← Decrypt($M$*, $K$)
4:  $F$ ← Decrypt(C, $Pk$)
6:  Output: $F$, $Pk$
```

The example shown in Tables 2 and 3 illustrates how the scheme operates. With the sequence $\{r_1, \ldots, r_8\}$ and key K {0,1,1,0,1,0,0,1}, Kr is $\{r'_1 = r_2, r'_2 = r_3, r'_3 = r_5, r'_4 = r_8\}$. The key K is recovered by comparing the Kr and the full sequence.

Table 2. Generation of an orderly sequence of responses from K.

Responses	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$	$\mathbf{r_5}$	$\mathbf{r_6}$	$\mathbf{r_7}$	$\mathbf{r_8}$
Key K	0	1	1	0	1	0	0	1
Subset Kr	-	$r'_1 = r_2$	$r'_2 = r_3$	-	$r'_3 = r_5$	-	-	$r'_4 = r_8$

Table 3. Key recovery from the orderly sequence of responses.

Subset Kr		$\mathbf{r'_1}$	$\mathbf{r'_2}$		$\mathbf{r'_3}$			$\mathbf{r'_4}$
Responses	$\mathrm{r_1}$	$r_2 = r'_1$	$r_3 = r'_2$	$\mathrm{r_4}$	$r_5 = r'_3$	$\mathrm{r_6}$	$\mathrm{r_7}$	$r_8 = r'_4$
Key K	0	1	1	0	1	0	0	1

4.2. *Example of Use Case in Zero-Trust Networks*

The protocol presented in Section 4.1 is applied to protect autonomous vehicles operating in a zero-trust network with poor signal quality [56–60]. For example, the vehicle contains a set of encrypted files with the instructions and software revisions required to react to certain circumstances. The set of keys required to decrypt these files should not be stored in the vehicle for security reasons; therefore, a server transmits them through the open network as requested. The latencies of the suggested protocols must be small for near real-time operations. The use of standardized cryptographic algorithms is preferred.

4.2.1. Initial Set Up—Enrollment

The enrollment cycle performed in a secure environment is summarized in Figure 6. The CRP processing, as described in Section 4.1, enables the encryption of each file with a secret key Sk, the generation of the two sub-keys Kc and Kr required to find the public key Pk. Let M* be the encrypted Pk with ephemeral key K. Let S be the seed randomly picked to generate the challenges. Let Kc be the stream { S, ω, M*}. Let Kr be the subset of responses computed from the full set and K.

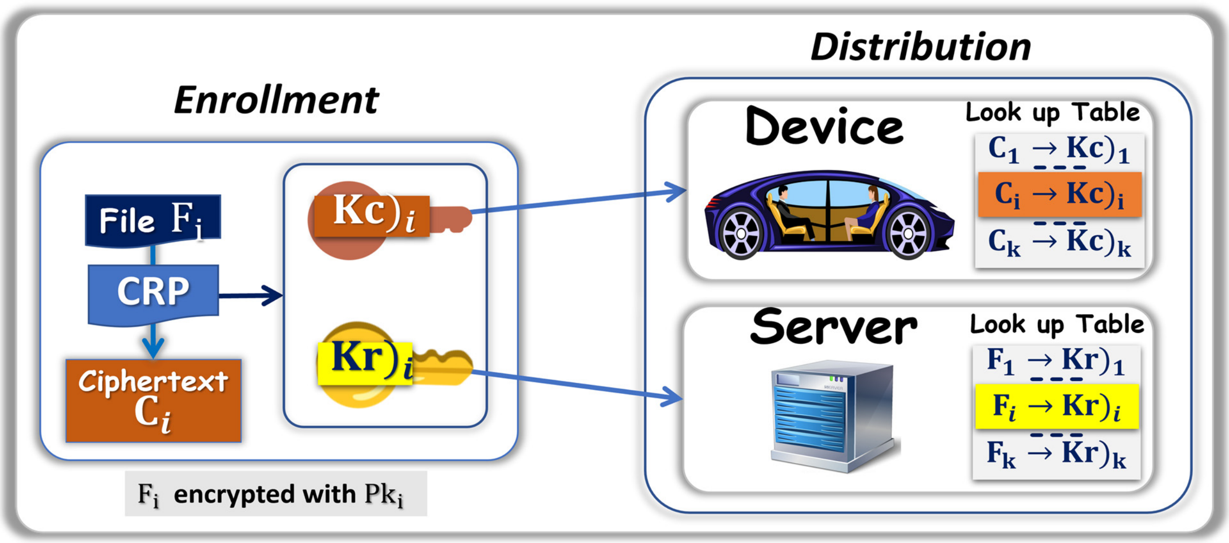

Figure 6. Block diagram of the enrollment cycle. From file F_1, two keys are computed by the CRP processing element: $Kr)_1$ from the responses and $Kc)_1$ from the challenges. F_1 and $Kr)_1$ are restricted by the server while C_1 and $Kc)_1$ are distributed to the terminal device.

After completion of the enrollment cycles, the server restricts all files Fs and associated subkeys Kr, while the autonomous vehicle restricts the ciphertexts C and subkeys Kc.

4.2.2. Verification of Authenticity—Deciphering the Files

If the vehicle encounters a problem, the engineers working remotely dictate that the vehicle should use file F_i; thus, they transmit $Kr)_i$ through the network (see Figure 7).

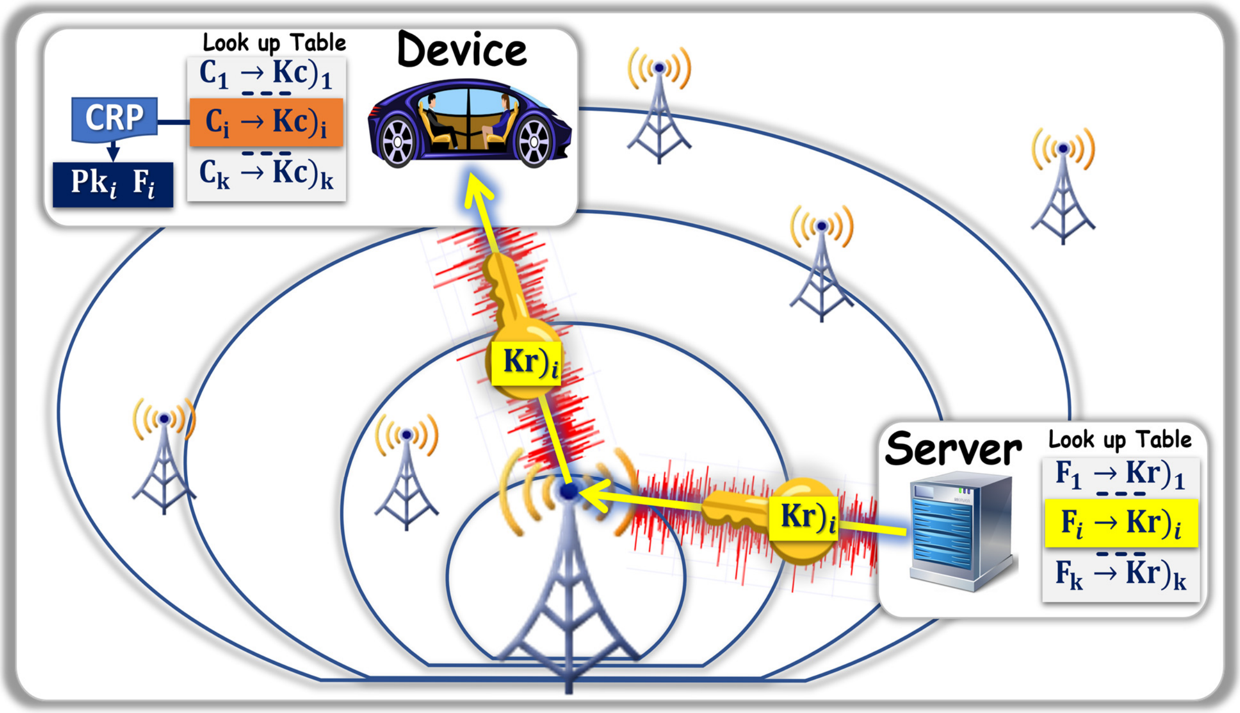

Figure 7. Block diagram of the recovery cycle. The server transmits $Kr)_1$ to the device in the noisy network. The CRP processing element of the terminal device is retrieving F_1 and the public key Pk_1 from keys $Kr)_1$ and $Kc)_1$. Verification of authenticity of F_1 is enabled.

Assuming that the rates of collisions between 100-bit long responses are negligeable, the vehicle can quickly determine which file to retrieve by testing a few responses. The protocol presented in Section 4.1 is able to manage poor signals and heavy-injected electromagnetic noises in the subkey $Kr)_i$ to retrieve F_i utilizing the following process:

- $Kc)_i$: $\{S_i, \omega_i, M^*_i\}$ enables the generation of the full set of responses from the ciphertext C_i, and the CRP mechanism.
- $Kr)_i$ discloses the subset of responses.
- Ephemeral key K_i is retrieved by comparing both sets of responses.
- Public key Pk_i is decrypted from M^*_i with key K_i.
- File F_i is decrypted from C_i with public key Pk_i.

If required, the noise can be directly injected into Kr by the server with a random number generator. The autonomous vehicle can also be equipped with the system to emit obfuscating noise during communication with the server, which has the potential to mitigate some side channel attacks. Having noisy responses can increase the one-wayness of the CRP mechanism by obfuscating the cryptoanalysis after a verification cycle. Without such a feature, the crypto-analyst is empowered to keep track of the CRPs for future analysis.

4.3. Security Analysis in Zero-Trust Networks

The protocols using a subset of responses, as presented above, require additional protections to mitigate certain vulnerabilities. A discussion of the selected suggested mitigations is discussed next. Other strategies are part of our proposed future work.

4.3.1. Loss of Both Kr and the Information Stored in the Terminal Device

A poor environment for this protocol is when Kr is intercepted as well as the content of the terminal device; hence, the opponent would then be enabled to retrieve the files. The structure of subkey $Kc)_i$: $\{S_i, \omega_i, M^*_i\}$ can enable MFA:

- MFA on the seed. The terminal device XORed the S_i with a password before storing it, then XORed it again during the recovery cycle. This can obfuscate the challenges generation process.
- MFA on the nonce: Rather than applying the MFA on the seed, the XORing can be performed on the nonce. This obfuscates the file used in the CRP mechanism.
- If the terminal device is driven by an operator, then biometry can be added.
- Additional protection of the public key Pk. Rather than having M* being the ciphertext of the public key Pk; the message M decrypted from M* during recovery does not directly disclose Pk. The recovery of Pk from M could also use MFA, or a separate code transmitted by the controlling server through a separate channel.

4.3.2. Verifying the Index of the Confidential File

Opening the wrong file may not be acceptable. To mitigate errors, an index can be added to the ephemeral key. The protocol is as follows:

- During enrollment, the index of 12 bits (when 4000 files are stored) is added in the front of each ephemeral key. The number of responses is extended to 12 + N. The number of responses of the subsets also increased, accordingly, by 12.
- The terminal device stores the 12 additional responses in a look-up table.
- During recovery, the first 12 responses of the subset are used to confirm the index from the first 12 responses pointing to the right ciphertext–Kc pair to decrypt.

During the recovery cycle, such a method can quickly verify the matching index. The key and the index are then linked to each other.

4.3.3. Replays, Man-in-the-Middle, and Side Channel Analysis

Both replays and the man-in-the-middle attack disturb normal operations and should be mitigated. The opponents intercept several subkeys Kr thus preventing the terminal device from receiving them. Subsequently, the opponent sends these subkeys, but not necessarily in the right order, which could have a catastrophic impact. The mitigation of such attacks add a feedback loop to inform the controlling server that the signal was well received by the terminal device at the right time. Adding a code at the end of each file that the terminal device has to send back to the controlling server is an effective implementation. Protection against the side channel analysis should be implemented. The protocol has several advantages from that standpoint: the noise injected in the network during the transmission of Kr can disturb the side channel analysis and the subkeys are only used once.

5. Statistical Analysis of the Protocols Based on Subsets of Responses and ECC

Practical issues are expected in the implementation of the methods described above in Section 4 since the noise injection could also generate erroneous bits in the cryptographic keys. One solution is to tolerate an approximate matching between the responses of the orderly subset generated during enrollment and the responses generated during verification. However, if the threshold of acceptance is too high, the probability of having two randomly chosen responses matching can also be too high. The term commonly used to describe such unwelcome matches is "collision" [61,62]. In order to optimize the protocols, we developed a predictive modeling of these collisions. We also developed error management schemes able to process small residual erratic bits.

5.1. Statistical Predictive Model for Collisions

The recovery of the file, as outlined in Algorithm 6, assumes that the noise injected in the P-bit long responses creates a BER lower than 25%, an arbitrary number that can be optimized. The number of erratic bits t is bounded below the threshold T, which in this case is given by $T = 0.25 \times P$. If the threshold is too high, then collisions between the N responses become a source of errors during the retrieval of key K. A collision occurs when two randomly picked P-long responses $\mathrm{r_a}$ and $\mathrm{r_b}$ share at least T bits with each other. We are assuming that all responses follow a binomial distribution; one in which q, the probability to obtain a state "0", equals the probability to obtain a state "1" ($q = 0.5$). We must ensure that the probabilities of having a collision Ψ(P,T,q) between the responses of the orderly subset and a randomly picked response are acceptably low. Thus, we developed a model to optimize the protocol that focuses on three variables of Algorithm 6: the rate of bad bits injected in the subset of responses, the number of bits P of the responses, and the number of bits of threshold T. This mode is created as follows:

- Assume that we have two P-bit long streams of responses Ra and Rb:

  Ra: $\{r_{a,1}, \ldots, r_{a,P}\}$; Rb: $\{r_{b,1}, \ldots, r_{b,P}\}$
- Each stream follows a binomial distribution.

  The parameter q is the probability q to have a 1.
- The two streams are XORed, bit by bit:

$$\{r_{a,1} \oplus r_{b,1},\ \ldots, r_{a,P} \oplus r_{b,P}\} \leftarrow \{r_{a,1}, \ldots, r_{a,P}\} \oplus \{r_{b,1}, \ldots, r_{b,P}\} \tag{3}$$

- The resulting stream also follows a binomial distribution with q = 0.5.
- The hamming distance H(Ra,Rb) between Ra and Rb is given by:

$$\mathrm{H(Ra, Rb)} = (r_{a,1} \oplus r_{b,1}) + \ldots + \left(r_{a,P} \oplus r_{b,)}\right) \tag{4}$$

- There is a collision when the hamming distance between $\mathrm{r_a}$ and $\mathrm{r_b}$ is: H(ra,rb) $\leq$ T.
- The rate of collisions is:

$$\Psi(\mathrm{P, T, q}) = \sum_{\mathrm{t}=0}^{\mathrm{T}} [\binom{\mathrm{P}}{\mathrm{t}} \mathrm{q^t (1-q)^{P-t}}] \tag{5}$$

The plot shown in Figure 8 was generated using T/P in the x-axis, and Ψ(P,T,q) in the y-axis. T/P varies between 0 and 40%, and Ψ(P,T,q) varies from 10^{-7} and 1. In our experiment, we measured the best fit for $q = 0.5$ since the distributions are almost perfectly binomial.

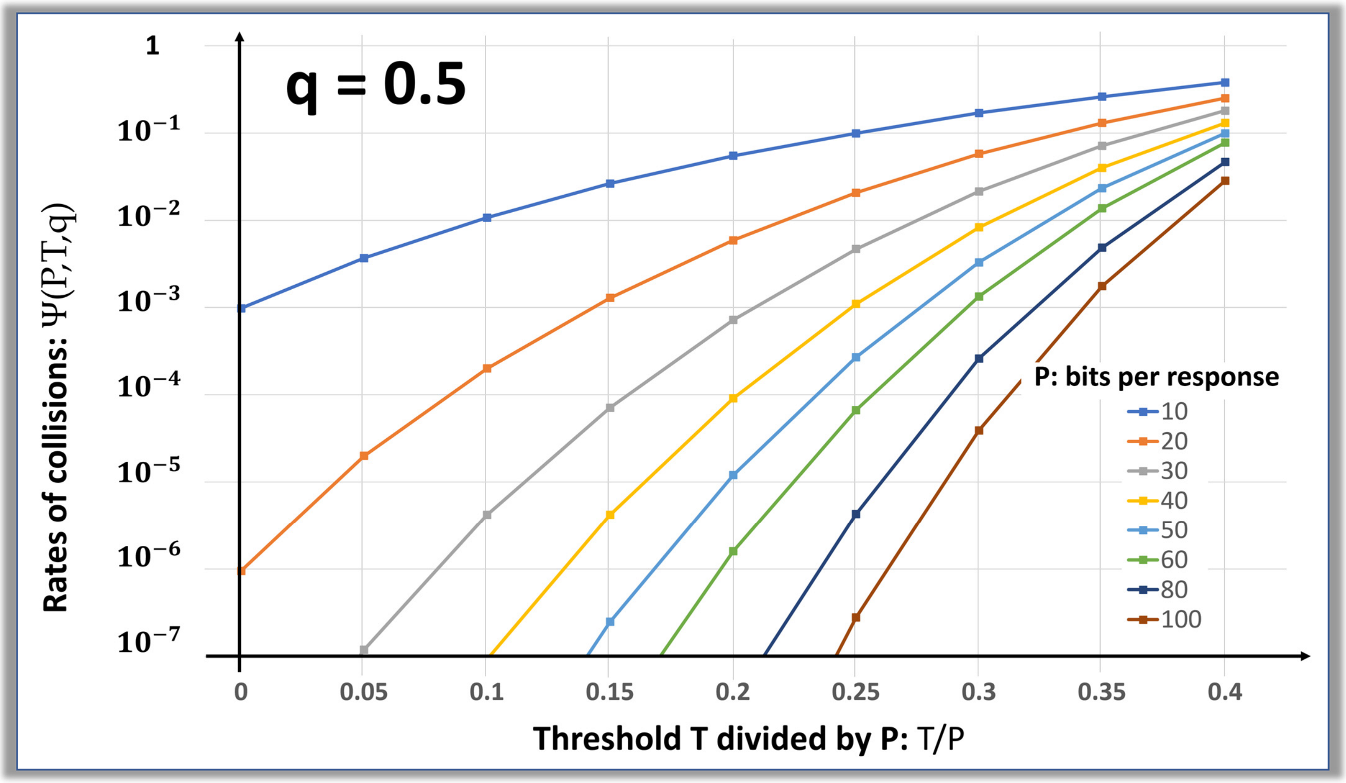

Figure 8. Modeling the rate of collisions as a function of threshold T, and P. To minimize collisions, T/P should not be set too high; however, the ratio should be set high enough to avoid errors due to the noise injection in the orderly subset of responses.

If we assume that there are no responses out of the f responses having the BER between enrollment and key recovery greater than the threshold T, the expected BER (E(K) in key K after recovery is expected to be given by:

$$\mathrm{E(K)} \;\approx \frac{f}{256} \times \gamma\, \Psi(\mathrm{P,T,q}) \;\approx 3\, \Psi(\mathrm{P,T,q}) \tag{6}$$

where γ is the number of times each response is randomly tested in the recovery cycle. As shown in Section 5.3.3, we used $\gamma = 6$.

If we assume that the BER of the noise injected in the responses are lower than $T = 0.25$, then an example of predictive use of the model for a 256-bit long key is as follows:

- $P = 40 \rightarrow \mathrm{E(K)} \approx 1.2 \times 10^{-3}$
- $P = 60 \rightarrow \mathrm{E(K)} \approx 6.7 \times 10^{-5}$
- $P = 80 \rightarrow \mathrm{E(K)} \approx 4.3 \times 10^{-6}$
- $P = 100 \rightarrow \mathrm{E(K)} \approx 2.8 \times 10^{-7}$

The BERs are exponentially reduced by increasing the length P of the responses. The light error correcting schemes, as presented below, quickly increase if the BERs are creating more than one bad bit in a 256-bit long key, which corresponds to a BER of 4×10^{-4}. This model predicts that $P = 80$ is acceptable and that $P = 100$ is a better operating point.

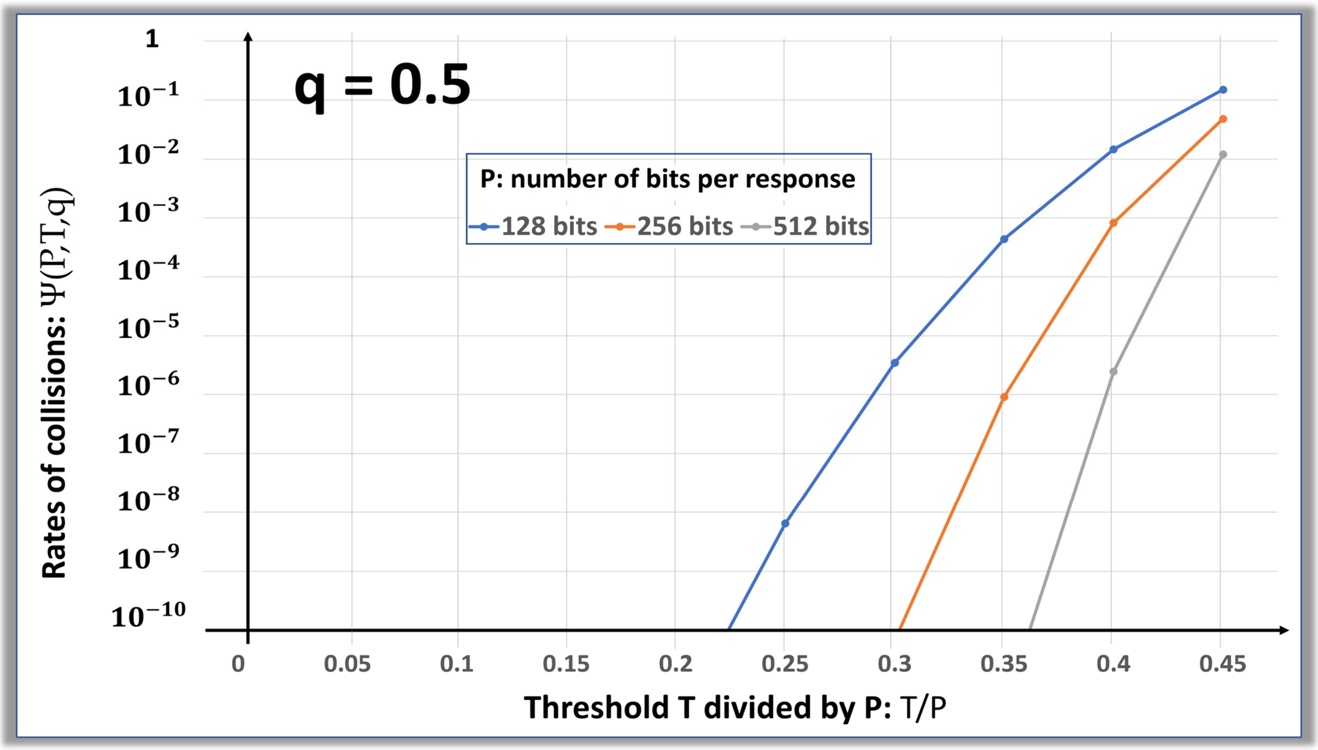

Figure 9. Modeling the rate of collisions with longer responses. With 512-bit long responses, noise injection techniques containing BERs as high as 40% in the subset of responses are considered since they only induce small rates of collisions (on the order of 10^{-6}).

5.2. Model for Heavy Noise Injections

The information presented in Figure 8 shows that responses in the range of 10 to 100 bits are too small to handle a noise injection of more than 30% erratic bits. A similar analysis is shown in Figure 9 with responses in the 128- to 512-bit range, and $q = 0.5$. An example of the predictive use yields the following results:

- $P = 128$ and $T = 30\% \rightarrow \text{E(K)} \approx 3.5 \times 10^{-6}$;
- $P = 256$ and $T = 35\% \rightarrow \text{E(K)} \approx 9.1 \times 10^{-7}$;
- $P = 512$ and $T = 40\% \rightarrow \text{E(K)} \approx 2.5 \times 10^{-6}$.

Such bit values of the error rates in K, which are in the part per million range (ppm), are widely acceptable for most applications. However, such an improvement introduces a negative impact on the latencies of the protocol, which are approximately proportional to the length of the responses. The bit error rates in the 35% range are large; thus, the penalty on latencies remains reasonable.

5.3. Error Management in the Cryptographic Keys

Mainstream ECC is difficult to implement in this approach since the data helpers needed to correct the key K are also affected by the noise injection. Thus, data helpers for the data helpers are called for, not an easy problem to resolve at high error rates. Therefore, we are proposing, in this section, several light error management schemes that can tolerate the residual errors created in K in the range of 1 to 3 bad bits per each 256-bit long key. With $T = 0.35$ and $P = 80$, 35% BERs in the responses generate $E(K) \approx 3.0 \times 10^{-4}$ BERs in the keys, which represents less than one bad bit per 256-bit long key.

5.3.1. Error Management with Response-Based Cryptography

The Response-Based Cryptographic (RBC) is a search mechanism allowing the recovery of a key when the ciphertext of the key is known [63,64]. In the protocol of Algorithms

6 and 7 with the subset of responses, the key K decrypts M* (the ciphertext of the key Pk) followed by the decryption of C and the ciphertext of the file F. This can be written as:

$$F = Decrypt(C, Pk) = Decrypt(C, Decrypt\,(M^*\,, K)) \tag{7}$$

[$Decrypt(C, Pk)$: decrypt "C" with key "Pk"]
Algorithm 8 verifies authenticity and decrypts Pk and F, see as follows:

Algorithm 8: Managing erratic key K with RBC

```
1:  Retrieve $K$ with CRP mechanism and from file $C$
2:  $x \leftarrow 0$
3:  Find all keys $\boldsymbol{Kn}$, $n \in \left\{1, \binom{N}{x}\right\}$, with hamming distance of $x$ from $K$
4:  Decrypt all files $Fn = Decrypt(C, Decrypt\,(M^*,Kn))$
    • If at least one of files $Fn$ is readable; $Pk = Decrypt(M^*,Kn)$; $F = Fn = Decrypt(C, Pk)$
    • Else:
        ○ If x $<$ 3: Increment $x$ by 1 and go back to step 3
        ○ If $x = 3$: Try again (go back to 1)
```

Using our computing system, we experimentally measured the throughput of the RBC at 2.0×10^8 cycles per second for AES and at 2.0×10^6 cycles/second for CRYSTALS-Dilithium. The code tested for Dilithium is freely available online, while AES is implemented in hardware within the computing element of our system [65]. The latencies to retrieve F through Algorithm 7 are listed in Table 4. The latencies to retrieve F are acceptable with AES, while managing 3 bad bits with Dilithium is slow. We anticipate that the hardware implementations of Dilithium will be available in the near future.

Table 4. Latencies to verify authenticity with RBC.

Number of Errors	**0**	**1**	**2**	**3**
AES—Latencies (s)	10^{-8}	2.5×10^{-6}	3.2×10^{-4}	2.2×10^{-3}
Dilithium—Latencies (s)	10^{-5}	2.5×10^{-3}	0.3	27

5.3.2. Error Management of Collisions

We developed and analyzed a method to experimentally detect collisions. The approach is based on the observation that when a collision occurs between a response from the subset and a response from the full set positioned in one of the states of "0" of K, that there is also a match with a response positioned in one of the states of "1" of K; this response from the subset then contains multiple matches.

Conversely, without collision, each response from the subset has one (and only one) match in the full set of responses. After experimentally finding the ρ positions of the subset with multiple matches, we conclude that all other positions (the number of which is f-ρ) are error-free. This significantly limits how many possible keys have to be tested in a methodology similar to that described by the RBC. For example: if one response of the subset sees two matches, then the first of the two can be a state of "1": while the second one is a state of "0" (and vice-versa). This leaves only two possible keys, which is quickly validated by computing F with Equation (7). If ρ responses from the subset create one collision each, then the number of possible keys per response is 2. This results in total 2^{ρ} possible keys required to be checked; a lower value than the number of cases checked by the brute force RBC for the same number of errors. The latencies for 256-bit long keys K are listed in Table 5.

Table 5. Latencies to find the correct key K with collision detection.

Number of Collisions ρ	**0**	**5**	**10**	**20**
AES—Latencies (s)	1.0×10^{-8}	3.2×10^{-7}	1.0×10^{-4}	1.0×10^{-2}
Dilithium—Latencies (s)	1.0×10^{-5}	3.2×10^{-5}	1.0×10^{-2}	10

5.3.3. Failure to Detect Matching Responses

Consider the case where the errors injected in the responses of the subset are greater than threshold T. In this case, ρ responses see no matches. This type of failure creates only a limited number of possible keys K. The positions in the sequence of orderly response matching with the responses-located positions (ρ-1) and (ρ-2) are known, and only a small number of positions with state "0" are between the two. The number of possible keys is then also small, quickly validated with Equation (7). The method to handle the collisions and failures to detect matching responses is described below in Algorithm 9:

- For each of the responses of the subset, at least γ_0 successive responses of the full set are tested for a potential match. Assume that key K has been tested to have no more than γ_0 successive 0s. This property must be validated during the random pick of K. If K has more than γ_0 successive 0s, then the key can be modified by inserting a few 1s to be sure that the condition is fulfilled. We picked $\gamma_0 = 6$ in our implementation.
- Detect the responses of the subset with either zero, one, or more than one matches with the responses of the full set of responses.
- List all possible keys.
- Verify the authenticity of F with all possible keys with Equation (7) of Section 5.3.1.

Algorithm 9: detecting and correcting errors during recovery of K

```
1: Enter the subset of response $\{r'_1, \ldots, r'_j, \ldots, r'_f\}$, the set $\{r_1, \ldots, r_i, \ldots, r_N\}$, $\gamma_0$ and $T$
2: Start with $j = 1$, $i = 1$, and $\gamma = \gamma_0$
3: While $i < N + 1$:
   Measure hamming distance $H(r'_j, r_{i+k})$ for all responses $r_{i+k}$ with $k \in \{0, \gamma - 1\}$
     • If $H(r'_j, r_{i+k}) \leq T$ for all $\gamma$ comparisons: Increment $i, j$ by 1, $\gamma = 2\gamma_0 - 1$
     [comment: to record that $r'_j$ has zero match]
     • If $H(r'_j, r_{i+k}) > T$ for only one position: Increment $j$ by 1, $i = i + k + 1$, $\gamma = 2\gamma_0$
     [comment: to record that position (i+k) is matching, and that $r'_j$ has only one match]
     • If $H(r'_j, r_{i+k}) > T$ for more than one position:
         ○ Find the position with the smallest number $k = k_{min}$, and with the largest $k = k_{max}$
         ○ Increment $j$ by 1, $i = i + k_{min} + 1$, $\gamma = \gamma_0 + (k_{max} - k_{min})$
     [comment: to record the list of all positions (i+k) matching with $r'_j$]
4: Find the correct key K
[comments: The positions $r'_j$ with only one matching response are considered correct. The positions $r'_j$ with zero match are caused by a failure to find the matching response. The positions $r'_j$ with two matching responses have at least 1 collision]
   4.1: List all possible keys Kn from the analysis
   4.2: Decrypt all files $Fn = Decrypt(C, Decrypt(M^*, Kn))$
   Find readable file $F = Fn$; then $Pk = Decrypt(M^*, Kn)$
```

Such a combined method will fail when both collision and the failure to detect occurs within the same response of the subset. When the BERs are in the 10^{-3} range, the probability of such an event is in the 10^{-6} range. In this configuration, the response in question matches with one, and only one, response of the full set of responses, as the key is unable to detect errors. One solution is to perform a brute force RBC search and in case of a failure to recover K occurs, request a new subset of response and retry.

6. Conclusions and Future Work

The suggested CRP mechanisms are directly using the digital files that must be protected and also their associated ciphertexts to generate ephemeral one-time use cryptographic keys. In these mechanisms, similarly with tracking blockchains, the digital files are hashed to be unique for non-repudiable and non-alterable operations. Two actual situations were discussed: (i) the encryption with proof of authenticity of the digital files distributed in storage nodes and (ii) the protection of the files kept in devices operating in zero-trust networks.

- In distributed networks, client devices provide the data needed for verification to agents driving smart contracts and storage nodes, the proposed protocols balance both security and transparency.
- In zero-trust networks, the sole priority is to enhance security and to protect the terminal devices. We developed a protocol allowing the injection of obfuscating noise in the keys transmitted by the controlling server to the exposed terminal device. BERS in the 25% range do not prevent the noisy key from decrypting the digital files stored in the device. To eliminate residual mismatches and generate error-free cryptographic keys, we developed and tested error management schemes to replace complex ECC, fuzzy extractors, and data helpers.

The continuation of this work will implement the algorithms for various applications to optimize security, reduce latencies, and add features specific to each application. We intend to perform an optimization with some of the following multi-factors:

- Length d of the digital file C* that is used for the CRP mechanism. Longer files are desirable to enhance randomness and minimize collisions. Excessive lengths negatively impact latencies.
- Number N of challenge–response pairs, and length P of each response. For the subset of response protocol, $N = 256$ or more is needed, and P has to be sufficiently long to avoid collisions, but not so long as to increase latencies.
- The acceptable BERs shall be injected in the subset of responses and threshold T shall be set at an acceptable level for the determination of a match.

In addition to the experimental work, we are currently developing a comprehensive multi-variate predictive tool. Additional attack vectors are also being investigated with various methods to inject noise and disturb the protocol under consideration. Lastly, the quality of true random number generation (TRNG) is critical to the levels of security of the suggested protocols as well as the actual randomness of the resulting streams of responses. We are performing statistically valid experiments to enable the quantification of randomness with tools developed by NIST, DieHarder, TestU01, and others.

Author Contributions: Conceptualization, B.C., M.H. and J.H.; methodology, B.C., M.H. and J.H.; software, B.C., C.P., M.H. and J.H.; validation, B.C.; formal analysis, B.C.; investigation, B.C.; resources, B.C.; data curation, B.C.; writing—original draft preparation, B.C.; writing—review and editing, B.C.; funding acquisition, B.C. All authors have read and agreed to the published version of the manuscript.

Funding: Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-23-2-0014. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Data Availability Statement: Not applicable.

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

Abbreviations

List of acronyms used in this paper:

Acronym	**Definition**
AES	Advanced Encryption Standard
BER	Bit Error Rate
CRA	Challenge Response Authorization
CRP	Challenge Response Pair
CRYSTALS	Cryptographic Suite for Algebraic Lattices
DSA	Digital Signature Algorithm
ECC	Error Correcting Code
MFA	Multi-Factor Authentication
NIST	National Institute of Standards and Technology
OSR	Orderly Subset of Responses
PKI	Public Key Infrastructure
PQC	Post Quantum Cryptography
PUF	Physical Unclonable Function
RBC	Response Based Cryptography
SHA	Standard Hash Algorithm
SHAKE	SHA algorithm and Keccack
TRNG	True Random Numbers Generators
XOF	Extended Output Function
XOR	Exclusive Or

References

1. Singh, M.; Pati, D. Countermeasures to Replay Attacks: A Review. *IETE Tech. Rev.* **2020**, *3*, 599–614. [CrossRef]
2. Conti, M.; Dragoni, N.; Lesyk, V. A Survey of Man In The Middle Attacks. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2027–2051. [CrossRef]
3. Uma, M.; Padmavathi, G. A Survey on Various Cyber Attacks and Their Classification. *Int. J. Netw. Secur.* **2013**, *15*, 390–396.
4. Vanstone, S.; van Oorschot, P. *An Introduction to Error Correcting Codes with Applications*; Springer International Series in Engineering and Computer Science Book 71; Springer: Berlin/Heidelberg, Germany, 2013.
5. Korenda, A.; Afghah, F.; Cambou, B. A Secret Key Generation Scheme for Internet of Things using Ternary-States ReRAM-based PUFs. In Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, Cyprus, 25–29 June 2018.
6. Darbon, j.; Sankur, B.; Maitre, H. Error correcting code performance for watermark protection. In *Security and Watermarking of Multimedia Contents III*; SPIE: Bellingham, WA, USA, 2021; Volume 4314. [CrossRef]
7. Gamage, H.; Weerasinghe, H.; Dias, N. A Survey on Blockchain Technology Concepts, Applications, and Issues. *SN Comput. Sci.* **2020**, *1*, 114. [CrossRef]
8. Fang, W.; Chen, W.; Zhang, W. Digital signature scheme for information non-repudiation in blockchain: A state of the art review. *J. Wirel. Commun. Netw.* **2020**, *2020*, 56. [CrossRef]
9. Guggenberger, T.; Schlatt, V.; Schmid, J.; Urbach, N. A Structured Overview of Attacks on Blockchain Systems. In Proceedings of the Twenty-fifth Pacific Asia Conference on Information Systems, Dubai, UAE, 12–14 July 2021.
10. Aggarwal, S.; Kumar, S. Attacks on blockchain. In *Advances in Computers*; Elsevier: Amsterdam, The Netherlands, 2021; Volume 121, pp. 399–410. [CrossRef]
11. Tomasin, S.; Zhang, H.; Chorti, A.; Poor, V. Challenge-Response Physical Layer Authentication Over Partially Controllable Channels. *IEEE Commun. Mag.* **2022**, *60*, 138–144. [CrossRef]
12. Smith, J.; Lingham, V.; Driscoll, J.; Fraser, I. Methods and Systems of Providing Verification of Information Using a Centralized or Distributed Ledger. U.S. Patent 10,558,974 B2, 11 February 2020.
13. Chow, A.; Chan, P.; Haldenby, P.; Lee, J. Document Tracking on a Distributed Ledger. U.S. Patent Application No. 2017/0048216 A1, 16 February 2017.
14. Zang, X.; Liu, C.; Chai, K.; Poslad, S. Challenge-Response Assisted Authorization Scheme for Data Access in Permissioned Blockchains. *Sensors* **2020**, *20*, 4681. [CrossRef]
15. Kaehler, A. Secure Exchange of Cryptographically Signed Records. U.S. Patent 11,044,101 B2, 22 June 2021.
16. Covaci, A.; Madeo, S.; Motylinski, P.; Vincent, S. System and Method for Authenticating Off-Chain Data Based on Proof Verification. U.S. Patent Application No. 2020/0322132 A1, 8 October 2020.
17. Uhr, J.; Hong, J.; Song, J. Tampering Verification System and Method for Financial Institution Certificates, Based on Blockchain. U.S. Patent Application No. 2021/0226804 A1, 22 July 2021.
18. Sheng, X.; McGuire, T.; Hromi, J.; Chawla, R. Computationally efficient transfer processing and auditing apparatuses, methods and systems. U.S. Patent Application No. 2017/0228731 A1, 10 August 2017.

19. Manian, Z.; Krishnan, R.; Sriram, S. Hybrid Blockchain. U.S. Patent Application No. 2017/0243193 A1, 24 August 2017.
20. Watanabe, H.; Akutsu, A.; Miyazaki, Y.; Nakadaira, A.; Fujimura, S. Contract Agreement Method, Agreement Verification Method, Contract Agreement System, Agreement Verification Device, Contract Agreement Program and Agreement Verification Program. U.S. Patent Application No. 2018/0205555 A1, 19 July 2018.
21. Harvey, A. Blockchain Enterprise Data Management. U.S. Patent Application No. 2019/0207750 A1, 4 July 2019.
22. Afghah, F.; Cambou, B. Authentication Based on a Challenge and a Response, a PUF and Machine Learning. U.S. Patent 10,469,273, 5 November 2019.
23. Cambou, B.; Gowanlock, M.; Heynssens, J.; Jain, S.; Philabaum, C.; Booher, D.; Burke, I.; Garrard, J.; Telesca, D.; Njilla, L. Securing Additive Manufacturing with Blockchains and Distributed PUFs. *Cryptography* **2020**, *4*, 17. [CrossRef]
24. Cambou, B. Secure Digital Signatures Using PUF Devices with Reduced Error Rates. U.S. Patent 11,271,759, 9 March 2022.
25. Cambou, B.; Telesca, D.; Jacinto, H. PUF-protected methods to generate session keys. In *Advances in Information and Communication, Proceedings of the 2022 Future of Information and Communication Conference (FICC), Volume 2*; Springer: Berlin/Heidelberg, Germany, 2022.
26. Cambou, B.; Jain, S. Key Recovery for Content Protection Using Ternary PUFs Designed with Pre-Formed ReRAM. *Appl. Sci.* **2022**, *12*, 1785. [CrossRef]
27. Haasnoot, E. Presentation attack detection and biometric recognition in a challenge-response formalism Erwin. *EURASIP J. Inf. Secur.* **2022**, *2022*, 5. [CrossRef]
28. Mohamed, M.; Shrestha, P.; Saxena, N. Challenge-response behavioral mobile authentication: A comparative study of graphical patterns and cognitive games. In Proceedings of the ACSAC'19: 2019 Annual Computer Security Applications Conference, San Juan, Puerto Rico, 9–13 December 2019.
29. Blom, R. Challenge-Response User Authentication. U.S. Patent 7,194,765 B2, 20 March 2007.
30. Song, J.; Noh, S.; Choi, J.; Yoon, H. A practical challenge-response authentication mechanism for a Programmable Logic Controller control system with one-time password in nuclear power plants. *Nucl. Eng. Technol.* **2019**, *51*, 1791–1798. [CrossRef]
31. Rhee, K.; Kwak, J.; Kim, S.; Won, D. Challenge-Response Based RFID Authentication Protocol for Distributed Database Environment. In *Security in Pervasive Computing*; Hutter, D., Ullmann, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3450, pp. 70–84.
32. NIST. Status Report on the Third Round of Post Quantum Cryptography Standardization Process. 2022. Available online: https://doi.org/10.6028/NIST.IR.8413 (accessed on 29 September 2022).
33. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS P), London, UK, 24–26 April 2018; pp. 353–367. [CrossRef]
34. Avanzi, R.; Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS-KYBER Algorithm Specifications and Supporting Documentation, 3rd Round Submission to the NIST's Post-Quantum Cryptography Standardization Process. 2020. Available online: https://csrc.nist.gov/projects/post-quantumcryptography/round-3-submissions (accessed on 29 September 2022).
35. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 238–268. [CrossRef]
36. Bai, S.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS-Dilithium: Algorithm Specifications and Supporting Documentation, Submission to NIST's PQC Standardization Process. Available online: https://github.com/pq-crystals/dilithium/tree/round3 (accessed on 8 February 2021).
37. Bernstein, D.; Brumley, B.; Chen, M.; Chuengsatiansup, C.; Lange, T.; Marotzke, A.; Peng, B.; Tuveri, N.; van Vredendaal, C.; Yang, B. NTRU Prime: Round 3, Submission to the NIST's Post-Quantum Cryptography Standardization Process. 2020. Available online: https://ntruprime.cr.yp.to/nist/ntruprime-20201007.pdf (accessed on 24 May 2023).
38. Chen, C.; Hoffstein, J.; Whyte, W.; Zhang, Z. *NIST PQ Submission: NTRU Encrypt a Lattice Based Encryption Algorithm, Submission to the NIST's Post-Quantum Cryptography Standardization Process*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2017.
39. Hoffstein, J.; Pipher, J.; Silverman, J. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory*; Buhler, J.P., Ed.; Springer: Berlin/Heidelberg, Germany, 1998; pp. 267–288.
40. Faugere, J.; Gauthier-Umana, V.; Otmani, A.; Perret, L.; Tillich, J. A distinguisher for high rate McEliece cryptosystems. In Proceedings of the IEEE Information Theory Workshop, Paraty, Brazil, 16–20 October 2011; pp. 282–286. [CrossRef]
41. Bernstein, D.; Hulsing, A.; Kolbl, S.; Niederhagen, R.; Rijneveld, J.; Schwabe, P. The SPHINCS+ signature framework. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS'19, New York, NY, USA, 11–15 November 2019; pp. 2129–2146. [CrossRef]
42. Prest, T.; Fouque, J.; Hoffstein, J.; Kirchner, P.; Lyubashevsky, V.; Pornin, T.; Ricosset, T.; Seiler, G.; Whyte, W.; Zhang, Z. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU; Round 3, NIST PQC Standardization Process. 2020. Available online: https://www.di.ens.fr/~prest/Publications/falcon.pdf (accessed on 29 September 2022).
43. Bertoni, G.; Daemen, J.; Peeters, M.; van Assche, G. The Keccak SHA-3 Submission. Submission to the NIST SHA-3 Competition (Round 3). 2011. Available online: http://keccak.noekeon.org/Keccak-submission-3.pdf (accessed on 14 January 2011).
44. National Institute of Standards and Technology. *Secure Hash Standard (SHS)*; NIST Federal Information Processing Standards Publication 180–4; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015. [CrossRef]

45. National Institute of Standards and Technology. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*; NIST Federal Information Processing Standards Publication 202; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015. [CrossRef]
46. Rahardja, U.; Kosasi, S.; Harahap, E.; Aini, Q. Authenticity of a Diploma Using the Blockchain Approach. *Int. J. Adv. Trends Comput. Sci. Eng.* **2020**, *9*, 250–256.
47. Qazi, M.; Kulkarni, D.; Nagori, M. Proof of Authenticity-Based Electronic Medical Records Storage on Blockchain. In *Smart Trends in Computing and Communications*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 297–306.
48. Bell, M.; Green, A.; Sheridan, J.; Collomosse, J.; Cooper, D.; Bui, T.; Thereaux, O.; Higgins, J. Underscoring archival authenticity with blockchain technology. *Insights* **2019**, *32*, 21. [CrossRef]
49. Shetty, S.; Red, V.; Kamhoua, C.; Kwiat, K.; Njilla, L. Data provenance assurance in the cloud using blockchain. In *Disruptive Technologies in Sensors and Sensor Systems*; SPIE: Bellingham, WA, USA, 2017; Volume 10206, p. 102060I. [CrossRef]
50. Feng, T.; Bhowmik, D. The multimedia blockchain: A distributed and tamper-proof media transaction framework. In Proceedings of the 22nd International Conference on Digital Signal Processing (DSP), London, UK, 23–25 August 2017.
51. Pappalardo, G.; Di Matteo, T.; Caldarelli, G. Blockchain inefficiency in the Bitcoin peers network. *EPJ Data Sci.* **2018**, *7*, 30. [CrossRef]
52. Uddin, M.; Stranieri, A.; Gondal, I.; Balasubramanian, V. A Decentralized Patient Agent Controlled Blockchain for Remote Patient Monitoring. In Proceedings of the 2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Barcelona, Spain, 21–23 October 2019; pp. 1–8. [CrossRef]
53. Liang, W.; Fan, Y.; Li, K.; Zhang, D.; Gaudiot, J. Secure Data Storage and Recovery in Industrial Blockchain Network Environments. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6543–6552. [CrossRef]
54. Mohanta, L.; Panda, S.; Jena, D. An Overview of Smart Contract and Use Cases in Blockchain Technology. In Proceedings of the 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10–12 July 2018; pp. 1–4. [CrossRef]
55. Bedi, P.; Gole, P.; Dhiman, S.; Gupta, N. Smart Contract based Central Sector Scheme of Scholarship for College and University Students. *Procedia Comput. Sci.* **2020**, *171*, 790–799. [CrossRef]
56. Kukkala, V.; Thiruloga, S.; Pasricha, S. Roadmap for Cybersecurity in Autonomous Vehicles. *IEEE Consum. Electron. Mag.* **2022**, *11*, 13–23. [CrossRef]
57. Li, S.; Iqbal, M.; Saxena, N. Future Industry Internet of Things with Zero-trust Security. *Inf. Syst. Front.* **2022**. [CrossRef]
58. Yang, D.; Zhao, Y.; Wu, K.; Guo, X.; Peng, H. An efficient authentication scheme based on Zero Trust for UAV swarm. In Proceedings of the 2021 International Conference on Networking and Network Applications (NaNA), Lijiang City, China, 29 October–1 November 2021; pp. 356–360. [CrossRef]
59. Blåberg, J. Zero Trust in Autonomous Vehicle Networks Utilizing Automotive Ethernet. Master's Thesis, Chalmers University of Technology/Department of Computer Science and Engineering, Gothenburg, Sweden, 2022.
60. Hurley, J. Zero-trust is not enough: Mitigating data repository breaches. In Proceedings of the ICCWS 2023 18th International Conference on Cyber Warfare and Security, Towson, MD, USA, 9–10 March 2023.
61. Bustio-Martínez, L.; Letras-Luna, M.; Cumplido, R.; Hernández-León, R.; Feregrino-Uribe, C.; Bande-Serrano, J. Using hashing and lexicographic order for Frequent Item-sets Mining on data streams. *J. Parallel Distrib. Comput.* **2019**, *125*, 58–71. [CrossRef]
62. Pupunwiwat, P.; Stantic, B. Minimizing collisions in RFID data streams using probabilistic Cluster-Based Technique. *Wirel. Netw.* **2013**, *19*, 689–703. [CrossRef]
63. Cambou, B. Unequally powered Cryptography with PUFs for networks of IoTs. In Proceedings of the IEEE Spring Simulation Conference, Tucson, AZ, USA, 29 April–2 May 2019.
64. Cambou, B.; Philabaum, C.; Booher, D.; Telesca, D. Response-Based Cryptographic Methods with Ternary Physical Unclonable Functions. In *Advances in Information and Communication, Proceedings of the 2019 Future of Information and Communication Conference (FICC), Volume 2*; Springer: Berlin/Heidelberg, Germany, 2019.
65. Mohd, B.; Hayajneh, T.; Vasilakos, A. A survey on lightweight block ciphers for low-resource devices: Comparative study and open issues. *J. Netw. Comput. Appl.* **2015**, *58*, 73–93. [CrossRef]