Development and Implementation of Novel Pair Value Encryption System

Rushil Mallarapu*

June 6, 2021

Abstract

Multiparty cryptography (MPC) is a cryptographic paradigm that seeks to have multiple, mutually distrustful parties compute a joint function of their inputs. Homomorphic encryption (HE) is a related, but distinct paradigm that seeks to allow arbitrary computations to be lifted over encrypted data. In both cases, general versions of such protocols are highly complex, requiring multiple rounds of communication, introducing implementation weaknesses and overheads which limit their practical efficacy. This research merges the insights behind these paradigms to develop what we term Pair Value Encryption (PVE), a lightweight protocol for performing arbitrary Boolean comparisons over encrypted data originating from potentially distrustful parties. We propose an abstract definition for PVE systems, demonstrate a protocol implementing this specification, and prove this protocol is both correct and secure. Python implementation and an example use case in patient matching in healthcare systems demonstrate both the low overhead and real-world applicability of this paradigm.

Keywords — Multiparty Communication, Homomorphic Encryption, RSA

1 Introduction

Cryptographic systems are designed to facilitate sensitive communication between secret agents. In the majority of standard practical protocols, such as RSA or Diffie-Hellman, these secret agents are mutually trustful, but are seeking to protect their communications from malicious external observers (Rivest 1). This means methods for communicating data in a secure fashion between agents are plentiful and well-developed. However, systems for allowing agents who are mutually distrustful – meaning that some of these agents have a dominant strategy of betraying their counterparts and stealing information – are relatively underdeveloped and complicated. Therefore, there is a need for understanding how to better design cryptographic systems that can connect decentralized parties and maximize total utility of a given scenario without avoiding unstable equilibria of trust.

One such family of systems is Multiparty Cryptography, or MPC (Halevi 2). In MPC, cryptographic systems seek to compute a joint function of the inputs of multiple distrustful parties such that no one party can discover the inputs of other parties. Moreover, it necessitates the assumption that while some parties may be malicious, in wanting to access data beyond their authorization, the exclusion of such parties from the protocol is untenable, which is reflective of real-world situations in which the inability to address hidden bad actors presents a serious security flaw. MPC is commonly proposed for utilization in voting systems,

On the other hand, Homomorphic Encryption (HE), looks at low-trust communication from a more type-theoretic background. Such systems attempt to allow for arbitrary binary circuits to be computed over encrypted data[1]. This allows multiple potential applications in healthcare, banking, finance, and science.

However, both these paradigms have suffered from a lack of efficient implementations and theoretical hurdles associated with their expansivity and generalizability – the difficulty of developing a robust MPC or HE system scales superlinearly with problem size. Thus, we introduce a new paradigm, called Pair Value Encryption, or PVE, which combines the key insights from both MPC and HE to offer a simple, generalizable framework for efficient low-trust cryptosystems. We report the development of the theory behind PVE, an algorithm fulfilling this paradigm, and a Python implementation thereof.

2 Theory

In this section, we describe the general PVE protocol, describe an algorithm achieving this protocol, prove that it provides the correct results, and finally, prove it it is as hard to break as RSA.

2.1 PVE Protocol

Pair value encryption derives from a biomimetic influence – the core idea behind the protocol reflects the behavior of protein folding. In biological systems, chains of amino acids fold and combine into proteins, which can recognize other substrates and proteins by way of steric (i.e. in real space) matching. However, the key

* *rushil.mallarapu@gmail.com*

[1]For readers with a type theory background, the notion of homomorphic encryption is similar to an "encryption" functor (or applicative functor) – monadic computation is technically more powerful than HE.

amino acid chain is obfuscated by the complex, often hard-to-predict process of protein folding[2]. Thus, the PVE protocol was derived to function on a similar basis – the encryption step should obfuscate plaintext values, but the comparison step should combine two matched ciphertext values to reduce to a simple Boolean variable.

Consider an arbitrary set of potentially distrustful agents, each of has a secret value that they want to compare in some Boolean fashion to those of other agents. Let X be the *problem space*, or the set of possible values that an agent might take. Moreover, let $\phi : (X, X) \mapsto \{0, 1\}$ be a binary comparison function over X. Let Z be the *plaintext space*, or set of values used to encode the problem set[3]. This also implies the existance of an invertable mapping $\gamma : X \mapsto Z$, converting values in the problem space to the plaintext space.

The pair value encryption protocol is a set of three functions:

1. $e : (Z, K) \mapsto Z$ – encrypt values, based on an encryption key
2. $c : (Z, Z) \mapsto \{0, 1\}$ – compare encrypted values
3. $d : (Z, K) \mapsto Z$ – decrypt values, based on a decryption key

In order to preserve the structure of the scenario as between X and ϕ, which depend on the use case, and Z, which depends on the implementation, we require that γ be constructed such that the following lemma hold true.

Lemma 2.1 (Preservation)**.** *For all $x, y \in X$, let $x' := \gamma(x)$ and $y' := \gamma(y)$. Then $c(e(x'), e(y')) \iff \phi(x, y)$.*

2.2 Algorithm

The implementation we present of the PVE system is based on the mathematics of RSA. First, we describe the initialization procedure for the algorithm, which selects the encryption and decryption keys.

Algorithm 1: PVE Setup

```
  Input: Two independent primes $(p,q)$
  Result: PVE modulus $(n)$, encryption key $(e)$,
          decryption key $(d)$
1 $n \leftarrow p \cdot q$;
2 $\phi(n) \leftarrow (p-1) \cdot (q-1)$;
3 for $i \leftarrow Z$ do
4     if $\gcd(i, \phi(n)) = 1$ then
5         $e \leftarrow i$;
6     end
7 end
8 $d \leftarrow e^{-1} \pmod{\phi(n)}$;
  Output: $(n, e, d)$
```

Note that a necessary postcondition of this PVE initialization step is that $de \equiv 1 \pmod{\phi(n)}$. Now, we provide algorithms for the encryption, comparison, and decryption procedure.

Algorithm 2: PVE Encryption

```
  Input: Problem text $(x \in X)$
  Data: PVE modulus $(n)$, encryption key $(e)$
  Result: Ciphertext $(z' \in Z)$
1 $z \leftarrow \gamma(x)$;
2 $z' \leftarrow z^e \pmod{n}$;
  Output: $z'$
```

Algorithm 3: PVE Comparison

```
  Input: Two ciphertexts $(z, z' \in Z)$
  Data: PVE modulus $(n)$
  Result: Comparison $(cmp \in \{0,1\})$
1 $cmp \leftarrow z \cdot z' \pmod{n}$;
  Output: $cmp$
```

Algorithm 4: PVE Decryption

```
  Input: Ciphertext $(z' \in Z)$, decryption key $(d)$
  Data: PVE modulus $(n)$
  Result: Problem text $(x \in X)$
1 $z \leftarrow z'^d \pmod{n}$;
2 $x \leftarrow \gamma^{-1}(z)$;
  Output: $x$
```

2.3 Properties

We now prove that the PVE algorithm will correctly compare and decrypt encrypted values.

Theorem 2.2 (PVE Decryption Correctness)**.** *Let $z \in Z$[4] be an plaintext value encrypted by the PVE system (n, e, d). Then, it holds that $z \equiv \text{Decrypt}(\text{Encrypt}(z)) \pmod{n}$.*

Proof. Let $z' \in Z$ be equal to Encrypt(z). Then, it holds by Algorithm 2 that $z' \equiv z^e \pmod{n}$. Applying Decrypt to both sides of this congruence as per Algorithm 4, we have that $z'^d \equiv z^{ed} \pmod{n}$. Thus, it remains to show that $z^{ed} \equiv z \pmod{n}$. We have that $de \equiv 1 \pmod{\phi(n)}$, which implies there exists some k for which $ed = 1 + k\phi(n)$. We then have $z^{ed} \equiv z^{k\phi(n)+1} \equiv z \cdot z^{k\phi(n)} \equiv z \pmod{n}$, with the last congruence following from Euler's theorem. It therefore holds that $z^{ed} \equiv z \pmod{n}$, thus completing the proof. $\square$

Theorem 2.3 (PVE Comparison Correctness)**.** *Let $x, y \in X$ be a pair of problem values to be encrypted and compared by the PVE system (n, e, d) and the comparison function ϕ. Further, let γ be chosen by the Preservation lemma as above. Then, $\phi(x, y) \iff \text{Compare}(\text{Encrypt}(\gamma(x)), \text{Encrypt}(\gamma(y)))$*

[2]Of interest in this analogy is the fact that the protein folding problem is NP-complete.

[3]Encrypted values are also in Z – this has the benefit of meaning neither a malicious agent nor an outside observer can distinguish unencrypted from encrypted data.

[4]We work directly from the plaintext space as γ allows free interconversion.

Proof. As per application of Algorithm 2, let $x' = \text{Encrypt}(\gamma(x)) \equiv \gamma(x)^e \pmod n$ and likewise for y'. We can apply the Preservation lemma as per the quotient group $\mathbb{Z}/n\mathbb{Z}$ to conclude that $(\gamma(x)\gamma(y) == 1) \equiv \phi(x, y) \pmod n$. It remains to be shown that the comparison of the encrypted values is also congruent to $\phi(x, y)$. Observe that $x' \cdot y' \equiv (\gamma(x)\gamma(y))^e \pmod n$. If $\phi(x, y) = 1$, then $\gamma(x)\gamma(y) \equiv 1 \pmod n$, and so $x'y' \equiv \phi(x, y) \pmod n$. If not, then $(\gamma(x)\gamma(y))^e \not\equiv 1 \pmod n$, and $x'y' \equiv \phi(x, y) \pmod n$, thus completing the proof. $\square$

Finally, we prove that the algorithm is equivalent in strength to RSA (Rivest 2).

Theorem 2.4 (PVE Strength)**.** *Let there be a malicious agent with access to a ciphertext value z' and the public section of the PVE system (n, e). Then, it is impossible for this agent to access the plaintext value z.*

Proof. Assume the agent can calculate z. This means the agent can solve the equation $z^e \equiv z' \pmod n$, which requires being able to either factor n into its prime components or compute the discrete logarithm. In the first case, this task is at least as hard as integer factorization, and in the second, this task is known to be impervious to conventional methods for large n. Thus, we have that the agent's task is impossible for appropriately large n, thus completing the proof. $\square$

3 Implementation

The PVE system described above was implemented using the Python programming language. In addition, there is an attached Jupyter Notebook which gives a real-world use case in matching patients to healthcare specialists. From this, it is clear that the PVE is far more straigtforward than MPC or HE, allowing for ease of implementation, which prevents coding bugs that might open potential security flaws. Moreover, the performance of the codebase is high, on-par with other Pythonic cryptographic libraries.

This implementation also emphasizes the importance of finding γ such that the preservation lemma holds. We include a code sample for how such a function can be constructed, but the benefit of this protocol is that the construction of γ can be tailored to the specifics of the scenario at hand. Note here a fundamental limitation in what PVE is capable of achieving – in initial studies, attempts were made to search for an algorithm that could preserve arbitrary inner products over encryption. However, any transformation which preserves all inner products must is orthogonal by definition, and orthogonal transformations over finite-dimensional vector spaces are trivially invertable, making such a scheme highly vulnerable.

Something of note is that, due to its venerability, RSA has attracted many efforts to break both its theoretical basis – by developing heuristic algorithms that can efficiently perform integer factorization or solve discrete logarithms – or its practical realization – finding vulnerabilities in published and popular cryptographic libraries and distributing knowledge of these vulnerabilities through the dark web[5]. Due to the similarity of RSA and our presented PVE implementation, PVE is technically vulnerable to the same risks.

However, the long-term usage of RSA is indicative of how reliable the system is on average, meaning PVE implementations based on RSA technology are likely to stay salient up until RSA itself is broken, which would practically occur after the advent of large-scale quantum computing technology. Moreover, as PVE is primarily a protocol and not necessarily tied to any implementation, a more long-term solution would be to reimplement this protocol either using elliptic-curve fields, or with quantum encryption techniques themselves. As such, PVE's potential pitfalls are both minor and easily remedied.

4 Conclusion

In summary, we have developed the pair value encryption protocol, an amalgamation of MPC and homomorphic encryption paradigms that addresses unique problem spaces in cryptographic communication. We exhibit an algorithm for this protocol, prove its correctness and security, and describe our results with a Python implementation. PVE presents a powerful paradigm for cryptographically secure healthcare, hiring, matching, and other low-trust, decentralized applications. The codebase for our implementaiton has been open-sourced to allow for public analysis. Future work will focus on extending this protocol with more secure algorithms and developing production libraries backing up demonstrable real-world use cases.

Works Cited

Halevi, Shai, et al. "Round-Optimal Secure Multi-Party Computation." *Journal of Cryptology*, vol. 34, no. 3, 2021.

Rivest, Ronald L., and Burt Kaliski. "RSA Problem." *MIT Laboratory for Computer Science*, 2003.

Tseng, Yi-Fan, et al. "Homomorphic Encryption Supporting Logical Operations." *Proceedings of the 2017 International Conference on Telecommunications and Communication Engineering - ICTCE '17*, 2017.

[5]Note that the dark web is distinct from the deep web; the latter is simply the set of web sites not indexed by any search engine, whereas the former is a subset of the deep web used for nefarious and illicit activity.