MUSIC360

A 360 DEGREES PERSPECTIVE ON THE VALUE OF MUSIC

Deliverable 2.6

Secure and trusted sharing of music data – version 2

Disclaimer

The Music360 project has received funding from the European Union's Horizon Europe research and innovation action under grant agreement number 101094872.

Version history

Ver.	Date	Comments/Changes	Author/Reviewer
0.1	10/06/2025	Initial version	Yulu Wang
0.2	20/06/2025	OAuth2 identity server	Ed Green
0.3	25/06/2025	Overall architecture	Jaap Gordijn
1.0	29/06/2025	Processed review comments	Jaap Gordjn

Project Acronym	Music360	
Project Title	360 DEGREES PERSPECTIVE ON THE VALUE OF MUSIC	
Project Number	101094872	
Instrument	Research and Innovation Action (RIA)	
Topic	HORIZON-CL2-2022-HERITAGE-01-05	
Project Start Date	01/03/2023	
Project Duration	36 months	
Work Package	WP2 - Standardized, trusted and unified collection of music metadata	
Task	T2.3b Designing and implementing security mechanisms for controlled data access	
Deliverable	D2.6. Secure and trusted sharing of music data – version 2	
Due Date	30/06/2025	
Submission Date	30/06/2025	
Dissemination Level[1]	Public	
Deliverable Responsible	VU	
Version	1.0	
Status	Final	
Author(s)	Yulu Wang	VU
	Ed Green	VU
	Jaap Gordijn	VU
Reviewer(s)	Roel Wieringa	TVE
	Sabine Oechsner	VU

[1] PU= Public, CO=Confidential, only for members of the consortium (including the Commission Services), CL=Classified, as referred to in Commission Decision 2001/844/EC

Disclaimer/Acknowledgement:

“The project Music360 has received funding from the European Union’s Horizon Europe research and innovation programme under grant agreement No 101094872.

Contents

1 Introduction

This deliverable presents the design of the security system for the Music360 platform. As with many deliverables of the Music360 project, this deliverable has two versions. The first version has the goal to define the security baseline. This version, the second version will focus on advanced use cases, specifically the case that data, or computations based on the data, can be done by untrusted parties.

This document presents the current overall architecture for the Music360 platform. Our security concern focus on how to enable secure, trusted, decentralized sharing of music-industry data among multi-party stakeholders including Collective Management Organizations (CMOs), venues, artists, and policymakers. Version 1 (D2.3) builds the foundational security mechanisms, like industry-standard authentication via OAuth 2.0/OpenID Connect and fine-grained data access through access control list and PostgreSQL row-level security. Version 2 faces the critical challenge of facilitating secure data collaboration in partially untrusted environments.

The core work lies in addressing scenarios where stakeholders require joint computation over sensitive data without mutual trust. Where version 1 operated under the assumption of trusted data processors, version 2 investigates privacy-preserving computation (PPC) techniques, notably secure multi-party computation (SMPC), to resolve previously unmet requirements (e.g., R17–R20, R24–R27). This evolution directly tackles the MoSCoW priority C5 ("Data sharing to untrusted environments"), enabling parties like competing venues to contribute confidential inputs (e.g., revenue figures, artist earnings) while cryptographically guaranteeing that no participant learns another's raw data.

To validate this approach, the deliverable focuses on two security scenarios: (1) calculating the 'Average Increased Revenue' of venues, where policymakers derive insights without accessing individual venue income data; (2) calculating 'Artists Average Earnings' across CMOs, which requires deduplicating artist records without revealing cross-CMO details. We proposed a structured solution design procedure and the (decision-making) technology selection framework. It bridges scenario-specific requirements to specific SMPC protocols, via evaluating general PPC technology features, like threat models, deployment architectures, and computational trade-offs. Proof-of-concept implementations demonstrate practical feasibility, using tools like Prio+ and MP-PSI to achieve confidentiality, correctness, and regulatory compliance.

The document begins with an overview of the Music360 architecture's security-related aspects (Section 2), followed by a gap analysis from version 1 to derive hard-to-solve requirements (Section 3). Subsequent sections detail the solution design framework for security scenarios need PPC technologies (Section 4), Music360 scenario experimental validation (Section 6), and reflections on following scenario experiments. Overall, this work addressed Music360's goal: find a way to realize secure and trusted data sharing for music value across the decentralized music ecosystem.

2 Overall Architecture

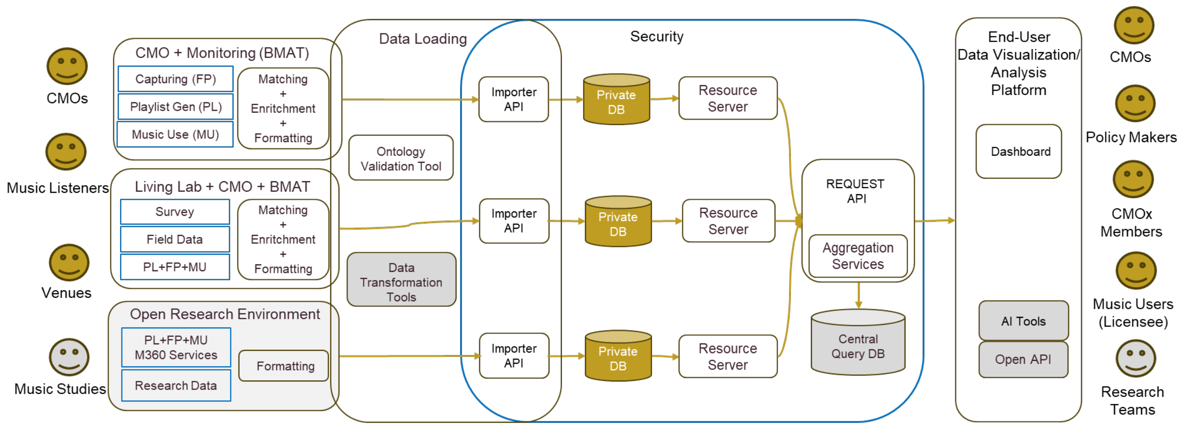


Figure 1 General data flow of Music360 architecture version 2 (in grey components under development)

The overall architecture of the MUSIC360 platform is presented in the perspective of data flows. It enables trusted, secure, and decentralized handling of sensitive music-related data among multiple stakeholders. The platform is designed for a multi-party environment, where each participant retains control over its own data while supporting secure, federated collaboration across organizations such as CMOs, CMO members, venues, and policymakers. The security mechanisms include authentication & authorization, access control, data isolation, privacy-preserving computation, and so on. These mechanisms are implemented consistently across the platform using well-defined protocols (OpenID Connect, OAuth2), modern database security features (PostgreSQL Row-Level Security), and token-based authentication (JWT).

Security considerations have influenced all layers of the MUSIC360 architecture, from backend services and APIs to the database and data import processes. A key architectural decision is decentralization, ensuring each data provider maintains ownership and operational control over its own partition of the distributed MUSIC360 database. This architecture guarantees data confidentiality, fine-grained access enforcement, and extensibility toward future users and providers beyond the project's initial consortium.

2.1 Authentication (identity management)

MUSIC360 supports a multi-party authentication model, enabling users to authenticate using credentials from multiple trusted identity providers. The authentication process works as follows:

- A user initiates login via the MUSIC360 frontend and selects an identity provider (e.g., neighboring rights CMO, author rights CMO).
- The identity provider authenticates the user and returns a signed authentication token.
- The data providers act as resource servers, and validate the token to grant access (or not).
- Identity information is exclusively stored by the identity providers and is not shared with other parties (i.e., the dashboard and aggregator) unless the user consents to it. This

information can be accessed in two distinct ways: inspection of the JWT claims assigned by the authorisation server or by REST API call to the resource server.
- Upon first login, users are prompted to link their rights roles (e.g., author and performer) to support data integrity in rights attribution.

This identity architecture ensures that users can seamlessly access MUSIC360 using decentralized login mechanisms, while the platform maintains a consistent, secure user management approach.

2.2 Authorization and access control

2.2.1 Token-based authorization

All access to data and services is governed by strict authorization policies, with MUSIC360 adopting OpenID Connect and OAuth2 protocols to manage secure login workflows and the issuance of authorization tokens. These tokens are implemented using JSON Web Tokens (JWTs), which encapsulate user-specific claims including identity and roles. Each backend component and the database is designed to verify the token’s authenticity and extract access rights from its payload, ensuring consistent and secure enforcement of access control across the platform.

2.2.2 Database-level security

MUSIC360 uses PostgreSQL’s Row-Level Security (RLS) mechanism to enforce fine-grained access restrictions directly at the row level within database tables. Each record includes an Access Control List (ACL), which specifies exactly which users or groups have what kind of permissions—such as read, write, or delete. When a JWT is received, its claims are decoded and mapped to PostgreSQL session variables that dynamically filter query results based on the authenticated user's rights. This ensures that access control is enforced uniformly and securely at the core of the data storage layer.

2.3 Data isolation and ownership

The Music360 platform employs a partitioned database architecture where each data provider maintains a private PostgreSQL database instance. These instances adhere to a shared Music360 Ontology V2 schema, ensuring consistent data representation for recordings, works, royalties, and usage metrics across all providers. Logical data isolation is enforced through provider-specific partitions, preventing unauthorized cross-provider access. Physical isolation is achieved using container-based deployments on AWS EC2 infrastructure, which separates hardware resources per provider.

While data owners may delegate administrative tasks to intermediaries such as CMOs, ultimate ownership and governance rights remain exclusively with the original provider. This includes full authority over data creation, modification, and deletion. The architecture allows partition overlaps where necessary, such as replicated recordings across multiple CMOs, while maintaining strict access boundaries through provider-specific query routing. Scalability is supported through horizontal data partitioning by provider or country, along with read-optimized replication strategies that align with the platform's primary read-heavy usage pattern.

2.4 Security layer components

The MUSIC360 platform incorporates a dedicated security layer composed of several coordinated elements. The Security Manager oversees all authentication and access control workflows across the system. Authentication is implemented using an open-source identity management solution, Keycloak, which acts as the Identity Server. Upon successful login, users receive a JWT containing verified identity claims and cryptographic signatures. Additionally, data providers may operate their own Authorization Servers, allowing them to act as both data custodians and identity authorities. This federated setup reinforces trust and supports distributed authentication workflows. All issued tokens are stored securely in client browsers and used in subsequent interactions, where backend components and databases validate their signatures to authorize access.

2.5 Secure data import and access flow

The data import process begins with the Importer API, which handles validation, transformation, and storage of data into the private database of each data provider. Every import task is executed within a transactional context and includes robust error handling to ensure data consistency. When users or systems make data access requests, their JWT tokens are authenticated and verified. These verified requests are then routed through backend components such as the Resource Server or Request API to the database, where access control is enforced using ACLs and RLS. This end-to-end process ensures that both the ingestion and retrieval of data are consistently secured across the architecture.

2.6 Privacy-preserving computation

To enable value extraction from sensitive music data while preserving privacy, the MUSIC360 platform allows data sharing in anonymized, aggregated, or derived forms rather than exposing raw, individual-level data. Users are able to conduct a predefined set of analytical operations, such as statistical aggregations, without gaining access to the underlying private information. As a proof of concept, the platform integrates Secure Multi-Party Computation (SMPC) and Private Set Intersection (PSI) technology in selected scenarios, enabling the computation of shared insights across multiple stakeholders without revealing any party’s confidential input. This aligns with MUSIC360’s core commitment to data sovereignty, trust preservation, and secure collaboration in untrusted environments.

3 Security Requirements

3.1 General (data) security requirements specifications

Appendix A summarizes and lists the (data) security requirements specifications about the Music360 platform. It is based on a series of workshops with the data owners, mainly the CMOs and BMAT. Table 4-1 shows the stakeholder analysis and prioritization result using the MoSCoW methodology (see detail in D2.5).

Concern	MoSCoW prioritisation	Description
C1	MUST	Multi-party
C2	MUST	Access control at instance level
C3	SHOULD	Access control at property level
C4	MUST	Access control delegation
C5	MUST	Data sharing to untrusted environments
C6	MUST	Trusted identity providers
C7	MUST	Platform parties can not access each other data without having the proper access controls
C8	MUST	Technical openness to other parties, support of open (de-facto) standards
C9	MUST	Scalable in terms of users
C10	MUST	Scalable in terms of create/read on data
C11	COULD	Scalable in terms of update/delete on data

Table 1 MUSIC360 MoSCoW feature prioritisation

3.2 Existing security design and implementation (version1 overview)

3.2.1 Data custody and database level security (C1-C4, C7, C9-C11)

Version 1 enforces that each data provider (CMOs, BMAT, etc.) retains exclusive custody of its own partitioned dataset (eg., R1, R11, R14–R15). Concretely, each provider hosts its MUSIC360 database partition, manages its own backups and replication, and configures RLS policies so that unauthorized rows are never returned, even if a backend service has a vulnerability (e.g., R3, R4, R6). Those RLS policies combine Domain-Based Access Control (DBAC), Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC) and Access Control Lists (ACLs) stored in ACL columns as a JSON structure. In practice, every incoming SQL query is transparently filtered: the DBMS evaluates the user's JWT-derived scope (issued by the provider's authorization server), the user's role (e.g., CMO membership), and any explicit ACL entries before returning data, ensuring fine-grained, database-enforced authorization. Version 1 assumes a trusted storage environment and focuses on RLS (e.g., R5, R7, R8, R16).

3.2.2 Authentication flow and token management (C6, C8)

All user authentication and identity management in Version 1 rely on OAuth 2.0 and OpenID Connect (e.g., R21–R23). Public clients (via the Dashboard) must use the authorization code flow with PKCE (proof key for code exchange) , generating a code_verifier and code_challenge to prevent interception, while confidential backend services (e.g., aggregation service, plugins) use the standard authorization code flow (without PKCE) and store a client_secret securely. Each access token is a JWT signed by the provider's asymmetric key (R22), embedding claims such as sub, aud, exp, and scope. Resource servers validate JWT signatures (using the provider's published JWKs), check exp, iss, aud, and then apply RLS + ACL filters to every request. Token revocation is propagated immediately via the central Message Bus (e.g., Kafka), so that once a data provider revokes a user's token, no further requests using that token succeed (e.g., R8, R11). Refresh tokens are supported to reduce repeated logins, but if revoked, the blacklist on the message bus ensures revocation takes effect before the JWT expires.

3.2.3 Platform and interservice communication security (C8)

Version 1 mandates that all inter-service communication mandates TLS 1.2+ (R12). API gateways enforce rate limiting, token validation, and attack mitigation (R13). The message bus enables near-real-time token blacklisting for revocation consistency (R11). Plugins (e.g., User/session management) delegate session integrity to the database, while aggregation services operate as trusted extensions of the Dashboard without token persistence.

3.2.4 Data governance design (C1, C7-C11)

Data providers retain full ownership and whitelist control over client access (R3). GDPR alignment includes user consent workflows and pre-sharing risk disclosures (R9, R10, R25). FAIR-compliant data exposure (R27) and partitioned storage ensure providers manage their datasets independently (R1, R2, R4).

3.2.5 Summary

As a summary, security design and implementations in D2.3 Version 1 has resolved critical requirements and addressed most of the stakeholder concerns through existing technologies like database-level controls we stated above:

Security design& implementation	**Requirements involved**	**Stakeholder concern involved**
Data custody & database security	R1, R3–R6, R11, R14–R16	C1-C4, C7, C9-C11
Authentication & token management	R8, R11, R21–R23	C6, C8
Platform & communication security	R12–R13	C8
Data governance	R1–R4, R7, R9–R10, R25, R27	C1, C7-C11

Table 2 Security requirement involved in Version 1

3.3 Hard-to-solve security issues

We need to focus on the remaining general (data) security requirements: mainly R17-R20, R24-R27 which can be identified as hard-to-solve security issues. We can generally derive these security requirements into several security scenario types below:

- **Secure multi-party data collaboration:** In multi-party computations, no single party can see all raw data. For example, R17 may forbid pooling sensitive data in one location, and R19 may require that inputs remain confidential even during joint analysis. Under C1 (e.g. "multi-party keep their own data ownerships") and C5 (limited trust among parties), the challenge is to compute global analysis without revealing any party's private inputs. Performance is also a concern since many parties may be involved.
- **Privacy-preserving data aggregation:** When releasing certain aggregation results, requirements like R24–R25 demand that individuals' privacy not be compromised. Simple aggregation (e.g. counting or averaging) may still leak sensitive details if adversaries have auxiliary knowledge. Under strict privacy regulations like minimize external data disclosure, we must publish useful data while provably preventing "reverse engineering" of individual sensitive information.
- **Encrypted Data Processing:** Some requirements (e.g. R18, R26) call for processing data without ever exposing it in plaintext. For example, R18 may require that sensitive data remain encrypted in transit, at rest, and even during computation. Under C8 (use of vetted cryptographic standards) and C5 (possibly limited on-device compute), the system must allow certain computations on data that stays encrypted. Traditional practice decrypts before compute (risking data exposure), which violates these requirements.

3.4 Scenario-specific security requirements

3.4.1 Scenario 1: Average increased revenue

3.4.1.1 Scenario description

In the MUSIC360 DBE, a policymaker would like to know the average increase in revenue of venues (located in a specific region) as a result of playing music in these venues. In order to use music, venues have to pay a fee to Collective Management Organizations (CMOs). Every country has at least one CMO, but usually there are more of them, each representing different kind of rightholders and intellectual property rights. CMOs have a mandate to collect fees for their rightholders. Revenue data of venues is confidential information, as it is competitive and sensitive data. E.g. if the CMOs can obtain the precise revenue increase number associated with music played at each venue separately, they may consider adjusting the venue's licensing fee accordingly. This could have a negative impact on venues. Therefore, we have the following expectations and assumptions:

- Only the venue itself should have access to its respective revenue increase.
- Policymakers, CMOs, and other venues should not have access to these specific revenue data or the identity of each venue to which they correspond.
- We assume that there is no trusted third party can serve as a aggregation/computation centre in this scenario.

3.4.1.2 Scenario requirements

For the scenario 'Average revenue of venues', the following requirements are detailed and identified:

A. **Functional:** A party (e.g., policymakers) wants to know the influence (in terms of average revenue increase for venues) of playing music at a number of venues:
   a) The ability for venues to provide their data to compute the average revenue for a set of venues.
   b) The ability to compute the average over the input parties' data.
   c) The computation should be performed within an acceptable amount of time (seconds preferred, minutes acceptable, hours not).
   d) The underlying MPC protocol used for the computations should be open source for the feasibility of the proof-of-concept experiment.
   e) Complex privacy computational burdens should not be placed on venues due to typically limited computational resources.

B. **Security:** The average should be calculated without disclosing the individual revenue data from venues, which can be detailed as follows:
   a) Policymakers as final result receivers should only have READ access to the final result. Venues that did provide input data should only have READ access to the final result and all access to their own input data.
   b) *Input* parties deliver their input without disclosing their full and readable input to a single party. Parties interested in the *result* of the calculation obtain the complete result without obtaining the full result of a single party. This implies that no single party knows the input of the individual venue, except the venue itself, and also no single party knows the result, except the receiver (the policymaker).
   c) Individual revenue values of venues must remain confidential and should not be inferable from the final output or any intermediate data.
   d) The deployed solution must be resilient to collusion, e.g., a coalition of venues and computation servers should not be able to reconstruct venue-specific data.
   e) The technology should include mechanisms to prevent malicious inputs to guarantee the correctness and effectiveness of the result, such as zero or extreme outliers that could distort the average, without requiring disclosure of actual input values.
   f) There is no trusted third party who can do the calculation on behalf of the party interested in the result.

3.4.2 Scenario 2: Artists average earnings

3.4.2.1 Scenario description

In this scenario, a policymaker is interested in understanding the average income of artists within Europe belonging to a specific genre. The required information can be derived from royalty data held by multiple European Collective Management Organizations (CMOs). However, a key challenge arises: individual artists may be affiliated with multiple CMOs, receiving separate royalty payments from each. To compute a correct average, it is necessary to:

- Sum the earnings of all unique artists in the specified genre;
- Count the number of unique artists to divide the total sum appropriately.

Due to potential overlaps, deduplication across CMOs is essential.

Furthermore, privacy concerns limit what can be shared. While CMOs managing thousands of artists may safely disclose aggregate sums, smaller CMOs (e.g., managing only a few artists in one specific genre) risk revealing sensitive information even at the aggregate level. Therefore, we have the following expectations and assumptions:

- The computation of total earnings and artist count must occur without exposing either individual or CMO-level data.
- CMOs are assumed to be mutually distrustful and unwilling to reveal their data to a centralized third party.
- Each artist is assumed to submit one royalty claim per CMO.

3.4.2.2 Scenario requirements

For the scenario 'Performers average earnings', the following requirements are detailed and identified:

A. **Functional:** A party (e.g., policymakers) wants to know the influence (in terms of average revenue increase for venues) of playing music at a number of venues:
   a) CMOs must submit genre-specific earnings data for their registered performers, in a confidential and privacy-preserving manner.
   b) The system must compute the average earnings of unique artists, accounting for artists affiliated with multiple CMOs.
   c) Deduplication must be performed securely across CMOs using techniques like MP-PSI, with no leakage of overlapping artist identities.
   d) The protocol should complete within a practical time frame (seconds preferred, minutes acceptable, hours not).
   e) The selected secure computation protocol should be open-source, publicly verifiable, and experiment available.
   f) The protocol must avoid placing heavy computational burdens on participating CMOs.

B. **Security:** The average should be calculated without disclosing the individual revenue data from venues, which can be detailed as follows:
   a) Only the policymaker (result parties) (or other related service subscribers) is authorized to access the final result. CMOs should not access each other's data or intermediate outputs.
   b) Individual artist earnings must remain confidential, not visible to other CMOs, servers, or the policymaker.
   c) Artist overlaps across CMOs must be resolved through techniques like MP-PSI without disclosing matching identities to any party.
   d) The protocol must resist collusion: even if a subset of CMOs or servers collude, they should not reconstruct any individual artist's earnings.

e) CMOs should not learn the number of artists or total earnings of other CMOs, protecting both small and large CMOs from indirect data disclosure.
f) The final result (average earning) must not allow **reverse inference** about any individual artist or CMO, even in extreme value conditions (e.g., outliers, small cohorts).
g) Range proofs or statistical checks should be used to reject anomalous inputs (e.g., extremely high values) without requiring decryption or disclosure.
h) Metadata (e.g., artist genre affiliation, nationality) should be pseudonymized and processed minimally, ensuring GDPR compliance.

4 Security design: technology selection framework

4.1 Privacy preserving computation

MUSIC360 require robust privacy measures due to its distributed, multi-stakeholder nature. Traditional centralized security approaches, what we have realized in D2.3 (version 1), such as authentication, access control, are often inadequate in these environments. These stakeholders, such as venues, artists, rightsholders, and policymakers, frequently need to analyze and share data across organizational and geographical boundaries. However, doing so without compromising sensitive business information or violating data protection regulations (such as the GDPR) remains a critical challenge. Hence, **privacy-preserving computation (PPC)** techniques have become central to enabling secure, trustworthy, and compliant collaboration among diverse stakeholders.

PPC refers to a family of computational techniques that allow meaningful data analysis and decision-making without exposing the raw, private data itself. These techniques aim to ensure that each stakeholder can participate in collaborative processes (such as joint analytics or federated intelligence) while retaining control over their own data. Common methods under PPC include:

- **Homomorphic Encryption (HE):** Enables certain operations to be performed on encrypted data, generating results that remain consistent after decryption.
- **Differential Privacy (DP):** Protects individual-level information by adding statistical noise to results, providing strong guarantees even in large-scale analyses.
- **Secure Multi-Party Computation (SMPC):** Allows multiple parties to jointly compute a function over their inputs without revealing them to each other.

Each method offers unique strengths. For example, DP is effective for large-scale statistical releases but introduces utility loss. HE offers strong cryptographic guarantees but is computationally intensive and less suited for interactive scenarios. SMPC enables rich interactive workflows but requires coordination and may introduce performance overhead depending on protocol complexity and network conditions.

In MUSIC360, the diversity of envisioned use cases shows that the choice and configuration of PPC methods cannot be generic. Instead, they must be grounded in scenario-specific requirements. To address this, we are conducting ongoing research aimed at developing a scenario-specific PPC method selection framework. This work focuses on systematically mapping scenario-specific requirements to the technical capabilities and assumptions of individual or hybrid PPC techniques. The goal is to enable:

- Transparent and justifiable selection of appropriate privacy-preserving methods for each collaboration context;
- Traceable linkage between stakeholder-level goals and technical design decisions;
- Modular extensibility, allowing new use cases to be supported without reengineering the entire architecture.

This approach also facilitates some trade-off analysis, supporting decisions about acceptable levels of privacy leakage, performance degradation, or communication overhead based on the constraints of the specific scenario. By embedding this method selection process within the security design of MUSIC360, we aim to strengthen the alignment between technical mechanisms and stakeholder expectations.

4.2 Secure multi-party computation

4.2.1 SMPC advantages

Within the broader field of PPC, **Secure Multi-Party Computation (SMPC)** stands out as particularly relevant to the needs of the MUSIC360 project. SMPC enables a set of independent parties to compute a joint function over their private inputs, such that each party learns only the final result. This makes it ideal for environments like MUSIC360, where sensitive data is distributed among multiple actors who may not fully trust each other, yet need to collaborate to extract mutual value.

Unlike centralized data aggregation, which requires trust in a single entity, SMPC allows data to remain decentralized and under the control of its owner. Hence, SMPC generally supports the following key scenario requirement types (RQTs) (we put the full list in Appendix B) identified in MUSIC360:

- **[RQT1]** Confidential multi-stakeholder computation: Stakeholders can engage in joint computations without disclosing raw input values to each other, preserving the confidentiality of sensitive data.
- **[RQT2]** Trust-limited collaboration: The computation framework must avoid reliance on centralized or trusted third parties, ensuring decentralized execution without exposing data.
- **[RQT3] R**espect for distributed data ownership: The security design must ensure that data can only be included in computations when explicitly authorized by each independent stakeholder, reinforcing legal compliance.
- **[RQT4]** Protection against aggregation leakage: When publishing aggregate statistics, the security deisgn must ensure that individual contributions cannot be inferred, even with auxiliary external knowledge.
- **[RQT5]** Controlled exposure of computation results: Published results must be processed to prevent reverse-engineering of private inputs, while still retaining analytical utility.
- **[RQT8]** Collusion-resistant protocol design: The security design must prevent coalitions of dishonest participants from inferring private data or manipulating computation outcomes.
- **[RQT9]** Resilience to insider threats: Computation must remain secure even when some participants are compromised or act maliciously, without degrading global guarantees.
- **[RQT10]** Input/result verifiability without data disclosure: Security design should provide verification for the precise of computation results without gaining access to intermediate states or private inputs.

- **[RQT11]** Resource-constrained environments: Resource-constrained environments: The security design must remain functional in environments with limited computational capacity. This motivates the choice of cryptographic protocols that minimize resource consumption while still offering acceptable performance and security guarantees. In such settings, we aim for a practical trade-off between computational efficiency and implementation overhead, rather than adopting the most performant or most secure protocols in isolation.

These requirements do not apply uniformly across all use cases in MUSIC360. In different scenarios, stakeholders may prioritize different aspects. For example, some applications may tolerate weaker privacy assumptions in exchange for better performance or simpler deployment, while others may require stronger guarantees due to the sensitivity of data or regulatory considerations. In practice, the trust model, expected number of parties, network environment, and performance constraints all affect the suitability of different SMPC approaches.

To better support design decisions in this context, we are currently working on identifying how different privacy-preserving computation methods, particularly SMPC techniques, align with the specific requirements of typical scenarios in MUSIC360. This involves mapping scenario specific characteristics (e.g., party numbers, trust level, expected interaction frequency, and acceptable overhead) to the features of various SMPC protocols (e.g., semi-honest setting, based on secret-sharing technology). The goal is to help select appropriate methods for each case, based on a clear understanding of typical security scenario characteristics and what each protocol offers and requires.

This work is still ongoing for the investigation in more scenarios and forms part of our broader effort to make the application of privacy-preserving computation in MUSIC360 more structured and transparent, so that the chosen methods can be clearly related to the practical needs and constraints of the stakeholders involved.

4.2.2 SMPC technology features

4.2.2.1 The supposed [TSM] threat and security model

One important consideration in selecting an SMPC protocol is the possible threats and security model that defines the adversary capabilities in the presence of certain types of attacks. Most SMPC literature distinguishes between honest participants and adversaries. Honest participants are defined as those who adhere strictly to the prescribed protocol, refraining from any attempts to extract information beyond their designated input data or the intended computational output. In contrast, adversaries are characterized by their potential to compromise the integrity or confidentiality of the MPC protocol. This may involve manipulating the computation to produce incorrect or inaccessible results or engaging in coordinated efforts to infer sensitive information by aggregating the data they acquire *(Wang, 2016)*. Based on the behaviour of the adversary, security models can be divided into semi-honest adversary model, malicious adversary model and covert adversary model *(Zhao, 2019)*.

In the MUSIC360 digital ecosystem, adopting the semi-honest or covert adversary model is more reasonable and applicable. In this context, the semi-honest model is appropriate when stakeholders (like venues and CMOs) are assumed to follow the protocol but may attempt to infer sensitive information, such as the precise revenue increase of individual venues. This model is efficient and would allow stakeholders to receive aggregated average revenue growth data without exposing sensitive venue-specific information. However, in scenarios with a higher risk of malicious behaviour, the covert adversary model provides stronger security guarantees. This is particularly important when stakeholders, such as CMOs, might attempt to access specific venue revenue data to adjust licensing fees. The covert adversary model deters such actions by introducing the risk of reputation which usually has high value in DBE environment.

4.2.2.2 The [CM] Computing model

The computing model focuses on how calculations are structured within the protocol, how specific computing tasks are transformed into a form suitable for secure processing, and the encryption technologies and security assumptions relied on. Different SMPC approaches leverage distinct cryptographic techniques, protocols include those based on homomorphic encryption (HE), secret sharing (SS), oblivious transfer (OT), garbled circuit (GC), and the hybrid approach *(Song, 2021; Zhang 2021; Han 2023)*. These approaches always come with trade-offs in computational complexity, communication efficiency, and suitability for various types of operations. Additionally, the communication overhead and computational complexity usually increase significantly as the level of the security model increases.

4.2.2.3 The [DM] deployment model

The deployment model focuses on how the protocol is implemented in the actual system and the network architecture, that is, how the participants are distributed, how they communicate, and whether auxiliary parties or servers are introduced. IEEE standards association has classified the SMPC deployment model into three types: (1) Server-side MPC: Data providers upload encrypted data to a set of non-colluding servers that perform computations. This ensures data confidentiality but relies on the trustworthiness of the servers. (2) Peer-to-peer MPC: Venues run MPC computations themselves, reducing trust dependencies but increasing computational costs. (3) Server-aided MPC: A hybrid approach where some computation nodes are maintained by data providers, while others are external servers that assist in intensive computations like Beaver triple generation.

4.2.2.4 The [SF] Supported function

Generally, the technical basis of different SMPC protocols determines the scope of functions and the number of participants they can efficiently support. Regardless of the technical basis, a common fact of the SMPC protocol is that the types of function operations that can be supported in theory may have a wide range, but due to various limitations such as computational complexity and communication volume, different types of SMPC protocols at the practical application level still have different advantages for functions that can be supported relatively efficiently. Additionally, for complex functions, it is usually necessary to combine multiple SMPC technologies (Hybrid) to obtain a possible balanced implementation solution

(Guo, 2021). The table 3 below provides some hints for practical screening SMPC protocols/tools when the specific calculation function requirements and the involved participants' number have been clarified in scenarios:

Technical basis	Participants Number	Suitable function	Protocol examples
Based on HE	2, n	Additive, Multiplicative (Based on HE level)	SEAL, HElib, Overdrive, HEAAN, FHEW, TFHE, Lattigo, cuFHE, FAMHE, FV-NFLib, PALISADE
Based on SS	> 2	Linear, Nonlinear	MP-SPDZ, Wysteria, PICCO, FRESCO, JIFF, MPyc, Prio, Prio+, Whisper, Sharemind, EasySMPC, MPyC
Based on OT	> 2	Comparison, Polynomial calculation	EMP-toolkit, LibOTe, MASCOT
Based on GC	2, n	Comparison, Additive	EMP-toolkit, Obliv-C, ObliVM, CBMC-GC, Frigate, TinyGarble
Based on Hybrid	2, n	Linear U Nonlinear	ABY, ABY^3, Motion, SCALE-MAMBA

Table 3 SMPC protocol comparison about several scenario features

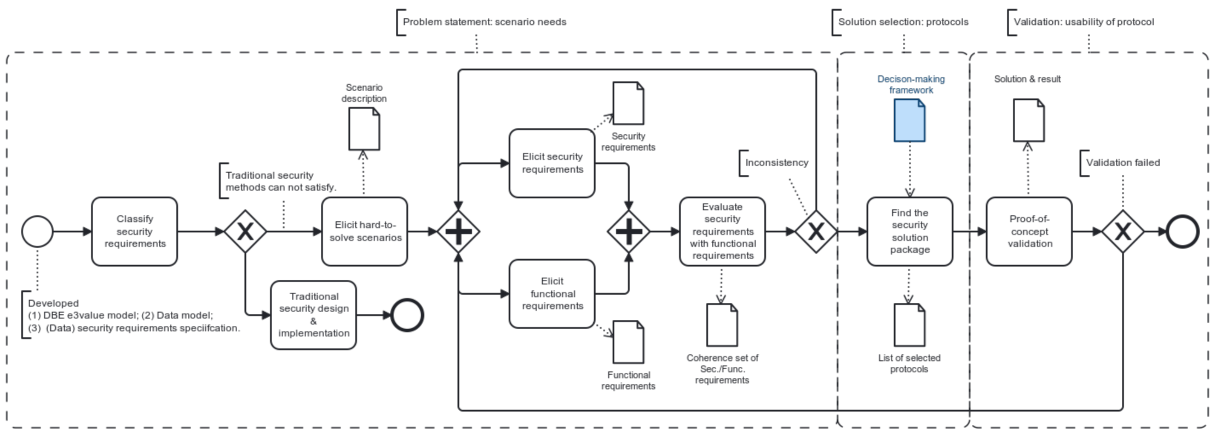
4.3 Private Set Intersection

Private Set Intersection (PSI) is a fundamental cryptographic primitive that allows two or more parties to compute the intersection of their respective private datasets without revealing any information about non-intersecting elements. PSI protocols have become increasingly important in privacy-preserving data collaboration, particularly in domains where sensitive attributes, such as identity, location, or income, must remain confidential *(Freedman, M. J., 2004; De Cristofaro, E., 2010)*. Classical PSI constructions rely on cryptographic tools like oblivious transfer, homomorphic encryption, or garbled circuits, but these often trade off efficiency for security. Recent advancements have led to highly efficient PSI implementations capable of handling large-scale datasets using OT-extension, elliptic curve techniques, or hybrid server-aided models *(Pinkas, B., 2014)*.

Applications of PSI span multiple domains, including privacy-preserving advertising attribution, medical data sharing, and fraud detection across institutions *(Dong, C., 2013)*. Despite these advancements, challenges remain in balancing malicious adversary resistance, low computational overhead, and support for real-world data variability.

For example, in our scenario concerning the average earnings of artists across different CMOs in specific genre, PSI enables multiple entities, such as several CMOs, to jointly compute overlapping artists identities without disclosing private financial records of non-overlapping artists. This approach ensures aggregate insights can be derived while preserving individual privacy and regulatory compliance.

4.4 Security solution design framework


Problem statement: scenario needs
Solution selection: protocols
Validation: usability of protocol
Classify security requirements
Traditional security methods can not satisfy.
Scenario description
Elicit hard-to-solve scenarios
Traditional security design & implementation
Developed
(1) DBE e3value model; (2) Data model;
(3) (Data) security requirements speciifcation.
Elicit security requirements
Security requirements
Elicit functional requirements
Functional requirements
Evaluate security requirements with functional requirements
Inconsistency
Coherence set of Sec./Func. requirements
Decison-making framework
Find the security solution package
List of selected protocols
Solution & result
Proof-of-concept validation
Validation failed

(1) Problem statement: Scenario needs

We explain this framework in the context of our Music360 DBE. The process begins with the e^3value model (see D 5.1), UML data model (see D 2.2), (data) security requirements specifications (in Appendix A). An e^3value model shows the objects of economic value that actors exchange, and because these objects are of value, they are vulnerable to attack. The data model defines the structure and relationship of data in the DBE, it ensures that the data is organized in a way that supports business needs and show some security scenarios. The (data) security requirements specification is analysed detailed in Sec 2. Requirement. Based on these artefacts, we introduce a classification step to identify hard-to-solve security scenarios and cluster security requirements accordingly. This step acknowledges that not all requirements can be addressed by traditional security methods. It can help us to jutisfy the necessity of applying expensive and complex PPC technologies in DBEs. We then elicit these specific security scenarios with their functional and security requirements. These are analysed for coherence and consistency, resulting in a final set of requirements, usually after a few iterations. Security requirements that can be solved by well-known techniques, such as traditional access controls, will not be further dealt with. These requirements should be satisfied for a DBE, but it is not our topic of interest.

(2) Solution selection: Protocol(s)

We then select the appropriate PPC techniques for implementing these security scenarios, using the proposed mapping framework (marked blue): constructed by two sides - feature extraction of known PPC techniques and realistic scenario-specific requirement characteristics. Given the maturity of the field of PPC, this extraction needs to be updated regularly by experts.

(3) Validation: Usability of protocol

A proof-of-concept prototype implementation of the target scenario through the selected PPC technique(s) will be conducted for validation in terms of requirement satisfaction.

4.5 Decision-making framework

We construct the decision-making framework by two main components: (1) Security patterns **Sec-pt** and their related requirement types **RQTs**, which can be linked to scenario-specific requirements items for reasoning; (2) PPC techniques features.

4.5 Framework application

4.5.1 Using the security solution design framework

We take the specific security scenario - Average increased revenue as an example to present the continued steps of security solution design framework process after derived the scenario-specific requirements. The requirements mentioned above cannot be solved by traditional security approaches tied to a single actor. Therefore, we analyze the `Average increased revenue' scenario in the SMPC context.

Involved parties

For the involved parties, we distinguish input parties, computation parties and result parties. Input parties provide the data needed for the computation, whereas the computational party performs the computations. Result parties are the intended recipients of the computation. Usually, in the MPC setting, there exist multiple computational models where each participant takes one of these three or a combination of these three roles.

- **Venues are input parties:** Each venue keeps its own increased revenue data. This input data is secret.
- **CMOs are computation parties:** CMOs receive and process the parts of the revenue data without being able to reconstruct individual revenues.
- **Policymakers are result parties:** Policymakers want to get the result: the average revenue increase.

Using the mapping framework

We have derived specific requirements for the scenario `Average increased revenue'. We linked the requirement types **[RQTs]** to the evaluated refined scenario-specific factors (by combining strong-related scenario requirement items) and mapped them to certain key PPC technology features via reasoning.

1. **Scenario factor: Required calculation type & function.**
    - *Requirement:* Type: sum (addition), mean (division).
    - *Reasoning:* Protocols supporting linear computation, especially an efficient addition operation, is needed.
    - *Related PPC feature:* **[SF]** Supported function: linear, addition.
2. **Scenario factor: Data ownership & confidentiality in multi-party nature. [RQT1, RQT2, RQT3, RQT4, RQT5]**
    - *Requirement:* Venues must retain their revenue data ownership and ensure this confidential data will not be disclosed.
    - *Reasoning:* Secret-sharing-based SMPC ensures raw revenue values are never reconstructed. Data is split into shares distributed across multiple servers, where no single server (or minority coalition) can infer private values.
    - *Related PPC feature:* **[CM]** Computing model: secret-sharing-based SMPC.
3. **Scenario factor: Collusion resistance. [RQT8, RQT9]**
    - *Requirement:* The main collusion risk exists between servers or venues and needs to be prevented.

- *Reasoning:*

  (1) In a contract-regulated business environment, contractual obligations between venues and CMOs can deter malicious behavior (e.g., submitting false data) to some extent. Hence, at least a semi-honest adversary model will be needed.

  (2) The primary risk is a passive inference of individual revenues, not active attack while secret sharing inherently mitigates this by splitting data into shares.

  (3) A server-side SMPC model creates a separation between computation servers and venues (clients), reducing collusion incentives to some extent.

- *Related PPC feature:* **[TSM]** Threat security model: at least a semi-honest adversary model; **[CM]** Computing model: secret-sharing-based SMPC; **[DM]** Deployment model: server-side SMPC model.

4. **Scenario factor: Resource-constrained.[RQT11]**
   - *Requirement:* Venues may lack computational resources to perform intensive cryptographic operations and communications locally.
   - *Reasoning:*

     (1) A server-side SMPC model offloads computation to dedicated servers. This eliminates the need for peer-to-peer coordination between venues, reducing local computational burdens and enabling the possibility of more participating venues.

     (2) We will prefer semi-honest rather than malicious-secure protocols to avoid incurring prohibitive overhead for frequent computations and communications.
   - *Rel*ated PPC feature: [DM] Deployment model: server-side SMPC model; [TSM] Threat security model: semi-honest adversary model.
5. **Scenario factor: Efficiency/effectiveness in computation task. [RQT10]**
   - *Requirement:* The computation of average increased revenue relies primarily on addition and mean operations. Mechanisms to prevent malicious inputs to guarantee the correctness and effectiveness of the result without requiring disclosure of actual input values.
   - *Reasoning:*

     (1) Secret-sharing protocols optimize arithmetic operations, outperforming Boolean-centric alternatives (e.g., garbled circuits) on most linear operations.

     (2) Server-side deployment usually minimizes latency for large-scale deployments.
   - *Related PPC feature:* **[DM]** Deployment model: server-side SMPC model; **[CM]** Computing model: secret-sharing-based SMPC.

In summary, our solution selection will be protocols supporting linear or addition operations, having a server-side deployment, using a secret-sharing-based approach, and under a semi-honest adversarial model.

4.5.2 Protocol selection

We take the specific security scenario – 'Average increased revenue' as an example to present this protocol selection process. The protocol selection is based on the scenario-specific technology feature preferences we reasoned from the mapping framework. As the first attempt, due to the time limit and page limit, we evaluate six protocols that are from active open source projects or have been widely discussed in the PPC community. Table 4 below shows how well these popular protocols meet these technical features and corresponding requirements.

SMPC technology	**[SF] linear, addition**	**[TSM] semi-honest**	**[DM] server-side**	**[CM] secret sharing**
Prio+ *(Addanki, S. 2022)*	●	●	●	●
Overdrive *(Keller, M., 2018)*	●	○	●	○
FRESCO (BGW) *(Amini, R., 2021)*	●	●	○	●
Mascot *(Keller, M., 2016)*	●	○	○	○
Whisper *(Rathee, M., 2024)*	●	○	●	●
EasySMPC *(Wieringa, R., 2012)*	●	●	●	●

Table 4 The degree of satisfaction of different protocols for scenario-specific technical

EasySMPC and Prio+ align well with all the SMPC technology feature preferences we reasoned from the mapping framework. Their open-source nature also shows their advantages in protocol selection. However, the feasibility of the experiment (even the real business analysis) also relies on the deployment difficulty and ease of use. The EasySMPC method utilizes email as the main digital communication channel, which may introduce new security problems and reduce reliability. Also, the data input approach in EasySMPC is more suitable for a certain amount of data that is well structured in CSV format, the approach is too heavy for our scenario needs. The Prio+ protocol is suitable for a proof-of-concept experiment because we can set different parameters according to a specific scenario to test its usability as a basic validation, and it ticks all the boxes. Hence, the Prio+ protocol was selected as a promising security solution for the scenario `Average increased revenue'.

6 Security Model and proof-of-concept experiments

6.1 Scenario 1: Average increased revenue

6.1.1 Security design & execution procedure

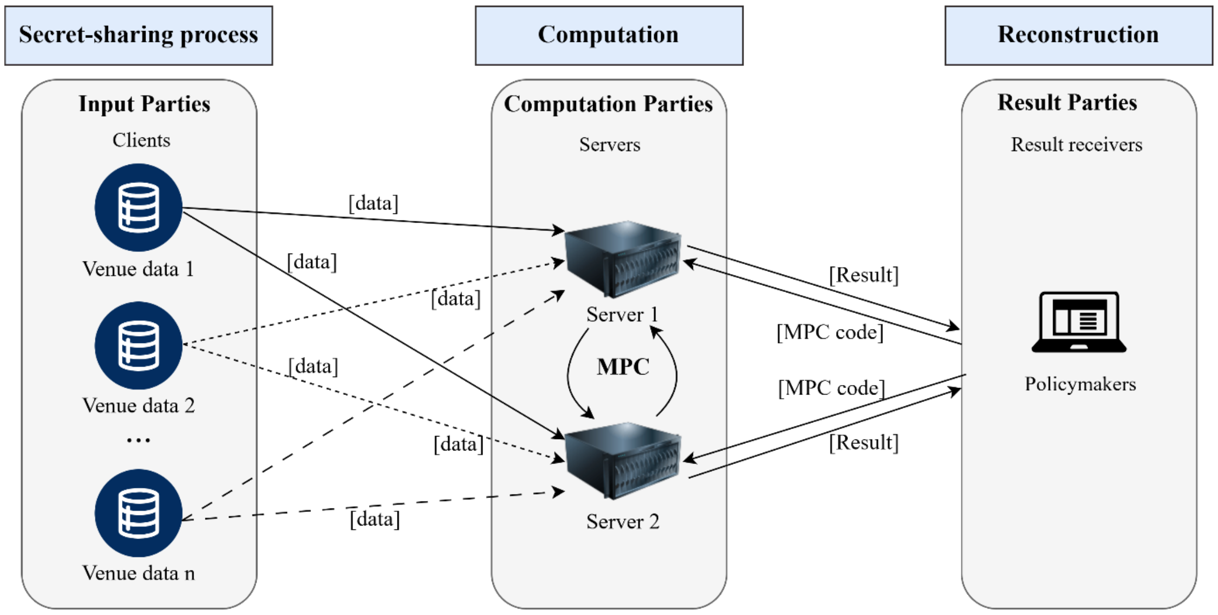


Figure 2 SMPC model for scenario 'Average increased revenue': [data] and [Result] are secret shares divided by SMPC protocol. Different line styles are used solely for visual clarity, without implying any semantic distinction.

Based on the above analysis, we listed relative results as below: (1) the selected PPC protocol: *Prio+*; (2) the required calculation formula: $R_m = \frac{\sum R_i}{n} \, (R_i = R_{i}\_before - R_{i}\_after)$; (3) the involved parties: input parties (venues), computation parties (CMOs), output parties (policymakers). All involved participants should acknowledge the specific computation task and the procedure of the security scenario, 'Average increased revenue'. The conceptual security model by the SMPC protocol for this scenario is shown in Figure 1. We explain it as below:

- **Input submission**: Each venue i has confidential revenue data: R_i and act as an input client to submit it. We assume that all venues will honestly submit their data since the selected protocol supports semi-honesty.
- **Data validation:** *Prio+* enables the input verification using share conversion to make sure the provided revenue increase data are within acceptable bounds (e.g., non-negative values or not excessively large).
- **Computation factor generation & aggregation:** Each input client generates multiple secret shares of its increased revenue data R_i. These shares are distributed to the aggregators.
- **Computation execution:** The computing servers collect the data shares from all the involved venues and execute computing tasks on their computing nodes according to the SMPC protocol.

- **Reconstruction & output:** The final sum result is sent back to the policymakers' clients and then reconstructed. The servers do not learn the individual inputs or the final result unless they collude. In *Prio+*, in order to compute the mean, the computation servers are allowed to modestly leak the number of involved venues and then clients can locally compute the mean.

6.1.2 Experiment setting & result

For the scenario proof-of-concept computation experiment, our basic container deployment using the *Prio+* protocol via Docker is conducted through simulating two server instances and multiple clients (operates in a cluster of individual clients). We used the supported function: *INT_SUM* because of the required computation type and function. We tried different values in two parameters: (1) *max_bits* (data size can be input): $2^{12}/2^{16}$; (2) *num_inputs* (number of venues): $10/100/1000$. These two parameter settings are related to the actual scenario nature. We look into two variables - total sent bits and total time - under the different settings of the two parameters we mentioned before. These two variables can help to measure whether the basic execution (time, data volume) of the selected protocol is within an acceptable range.

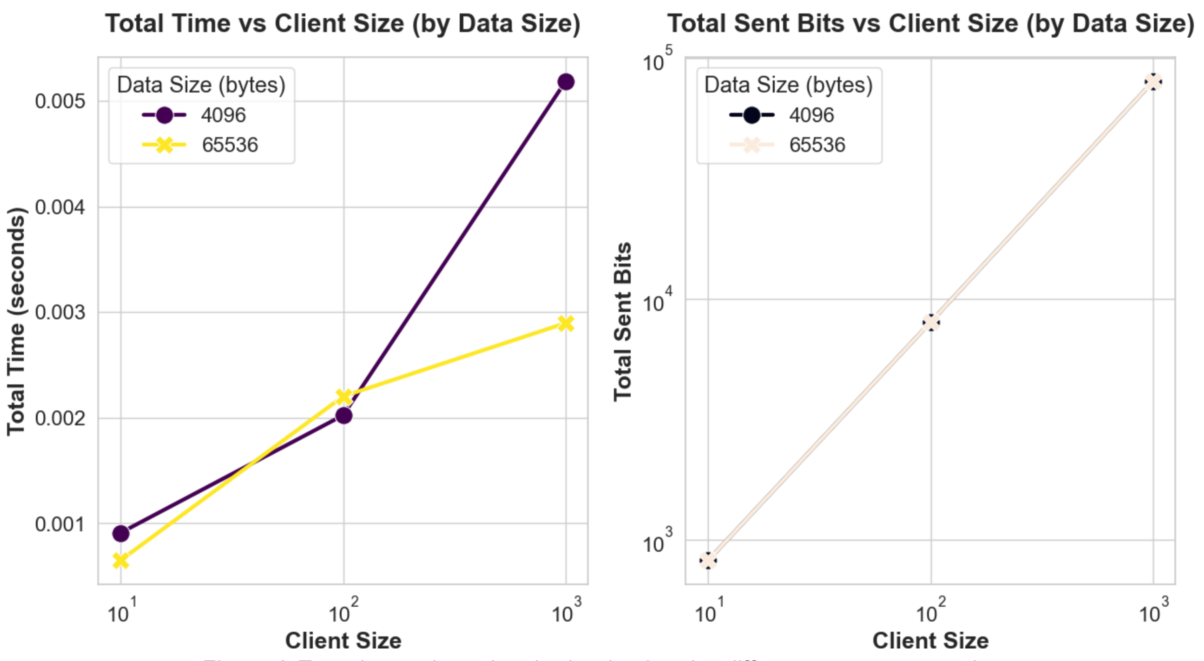


Figure 3 Experimental results obtained using the different parameter settings

6.2 Scenario 2: Artists average earnings

6.2.1 Security design & execution procedure

Based on the above analysis, we describe the secure computation process for deduplicated artist earnings aggregation among CMOs as below:

(1) the selected privacy-preserving protocols:

- PSI phase: Multi-Party Private Set Union using *Openminded PSI (Python)* or *EMP-PSI (C++)*
- SMPC phase: Secure aggregation using *MP-SPDZ (Python-like DSL)*, *FRESCO (Java)*, or *EMP-toolkit (C++)*

(2) the required computation formulas:

- Overall function: $\mu = \frac{S}{N} = \frac{\sum_{i=1}^{k} \sum_{a \in A_i} e_a^i}{|\bigcup_{i=1}^{k} A_i|}$
- μ: The average earning is the sum of total earnings divided by the total number of unique artists; S: The total earnings of each artist is the sum of their earnings in all associated CMOs. Assuming there are k CMOs, the set of artists in the ith CMO is A_i, and the earning of the artist a in the ith CMO is e_a^i; N: The total number of unique artists is the union size of all CMO artist sets.

(3) the involved parties:

- Input parties: Collective Management Organizations (CMOs), each holding private data entries $[artist\_id, earnings]$.
- Computation parties: CMOs acting as secure computing nodes.
- Output parties: Participating CMOs or designated authorities receiving the final result.

All participants must acknowledge the secure computing task and the associated privacy-preserving steps, aiming to achieve the final target: “Average earnings per unique artist”. The conceptual security model is explained as follows:

- **Input submission:** Each CMO i holds a local dataset of artist earnings in the form $[artist\_id, earnings]$, which is kept strictly confidential and never shared in plaintext with other CMOs. These datasets act as input to the secure protocols.
- **MP-PSI phase:** A privacy-preserving set operation is conducted among all CMOs to deduplicate the artist list. The objective in this phase is identify the *union* of artist IDs across all CMOs while no CMO learns which other CMO contributed which artist ID. Within the PSI output, each artist can be tagged with a private indicator of uniqueness (i.e., appears only in one CMO or in multiple). Protocol like Openminded PSI (Python) or EMP-PSI (C++) can be applied in this phase.
- **SMPC (secure multi-party computation) phase:** Once a deduplicated artist set is established, SMPC is used to compute the secure computation: (1) **Total earnings:** for each artist ID in the union set, participating CMOs contribute encrypted or shared earnings values (zero if not known), which are aggregated securely without revealing individual values. (2) **Unique artist count:** the size of the union set is computed securely. (3) **Average calculation:** final average earnings per unique artist is computed inside the SMPC protocol using secure division.
- **Execution & Output:** SMPC protocols (e.g., MP-SPDZ or FRESCO) can be applied to coordinate among computing parties (CMOs) to process shared values and compute the result. The final result (total earnings, unique artist count, average earnings) is revealed to output parties in plaintext, without revealing any individual artist data or CMO-level contributions.

This secure two-phase pipeline (PSI + SMPC) ensures privacy for all involved stakeholders while allowing collaborative analytics over sensitive artist earning data.

6.2.2 Experiment data setting

To evaluate the feasibility and performance of the proposed privacy-preserving computation pipeline (consisting of Multi-Party PSI and SMPC), we will do a proof-of-concept experiment for this scenario. For **participants and data distribution,** we assume five CMOs participate in the experiment. Each CMO holds a dataset containing a number of artist records in the form: $[artist\_id, earnings]$ The number of artists per CMO varies randomly between 2 and 300, representing realistic data volume differences across organizations. For example, one CMO might have only 2 artists, while another may manage 134 or 289. Artist IDs are simulated as anonymized identifiers (e.g., hashed integers), and importantly, partial overlaps are introduced across CMOs to reflect the real-world scenario where artists may receive earnings from multiple organizations. This ensures that the union set of all artist IDs is smaller than the total count across all CMOs, enabling a meaningful deduplication process in the PSI phase. It is an ongoing work and the realistic experiment process will follow the execution procedure we described above.

7 Reflections

Throughout the construction and evaluation of the early-stage solution design process and decision-making framework in the music DBE context, several key insights have emerged that highlight the limitations of our current approach.

- Practical constraints as another dimension

Practical constraints(e.g., such as cost, trust assumptions, regulatory compliance, and limited computational resources) have a decisive impact on SMPC deployment strategy. In real-world use cases like MUSIC360, participants such as small venues cannot afford high-latency or high-overhead protocols. Our framework allows trade-offs between security and performance but does not yet formally model or quantify these trade-offs. We suggest incorporating a resource-evaluation layer that explicitly measures protocol feasibility in constrained environments.

- Priority weighting between factors/features

The use of scenario factors (e.g., `resource-constrained', `collusion-resistant') allows qualitative prioritization. However, these mappings are currently performed manually and only rely on explanations as reasoning process. The framework lacks formal weighting mechanisms to balance competing requirements (e.g., performance vs. cost). A possible extension is to support multi-criteria decision analysis (MCDA) for feature ranking and protocol scoring.

- Gaps in scenario-to-requirement mapping

Although our framework maps functional and security requirements to scenario-specific factors, our experiment exposed cases where requirement types (RQTs) were underspecified or unmapped: (1) The scenario factor "required computation type" (e.g., summation + division) was difficult to trace to a specific RQT. This weakens traceability and hinders automatic reasoning. (2) The scenario mandated open-source protocol availability, yet this constraint was not represented in the framework. This suggests the need to add realistic, non-functional requirement types such as transparency or maintainability. These gaps indicate that the current framework may benefit from an additional component, potentially a non-functional requirement module for context-specific factors.

- Deployment complexity as an overlooked metric

Our experience shows that while many SMPC protocols align well with security and functional requirements in theory, deployment complexity often becomes the real challenge in practice. Even relatively practical options like Prio+ demand container orchestration and manual client-server coordination if needed. This indicates that deployment feasibility is multifaceted, involving aspects such as technical setup burden, integration cost, tooling maturity, and operational risk. We believe that future versions of the framework should include a Deployment Complexity Evaluation (DCE) component, to ensure protocol selection is guided not only by security alignment, but also by practical feasibility, especially in resource-constrained DBEs where technical capacity is limited.

- Need for broader types of RQTs.

The current list of 12 RQTs (grouped under six security patterns) was sufficient for the "average increased revenue" scenario. However, more complex scenarios in DBEs may have more security concerns. The MUSIC360 project itself includes other data flows (e.g., artist metadata, cross-country licensing) that raise such concerns. This suggests that our framework needs a broader and evolving taxonomy of requirement types, can ideally be expanded with new domain contexts.

8 Conclusion

This deliverable presents the security concern for the Music360 platform, focusing on secure and trusted sharing of music data. We apply advanced privacy-preserving computation techniques to enable secure data collaboration in environment without trusted third party. Building upon version 1 (D2.3), which established foundational security controls (e.g., OAuth 2.0/OpenID Connect authentication, access control mechanism, PostgreSQL RLS), Version 2 addresses the critical challenge of trust-limited multi-party computations**.** The core work include:

1. **Security requirement analysis**: Through analysis of security requirements specifications (Appendix A), we identified unresolved priorities from Version 1, notably **C5 ("Data sharing to untrusted environments")** and hard-to-solve requirements **R17–R20, R24–R27**. These gaps demanded advanced PPC solutions beyond traditional access controls, leading to our focus on security design for these specific security scenarios.
2. **Scenario-driven security design**: We propose a structured solution design procedure with the decision-making framework which maps scenario-specific requirements to SMPC technology features (e.g., threat models, deployment model), enabling protocol selection (e.g., server-side secret-sharing for resource-constrained venues).
3. **Apply SMPC technologies:** Secure Multi-Party Computation (SMPC) protocols (e.g., Prio+) were implemented in a proof-of-concept experiment to support scenarios like 'Average Increased Revenue', ensuring raw data confidentiality while allowing joint computation.
4. **Validation via PoC**: Proof-of-concept experiments demonstrated feasibility:
    - Scenario 1 achieved venue revenue average with no individual data disclosure.
    - Scenario 2 is an ongoing work, we have designed a hybrid PSI-SMPC pipeline for artist deduplication and earnings averaging.
5. **Data ownership preservation**: The current architecture enforces decentralized data ownership, allowing CMOs/venues full control over their partitions while supporting GDPR-compliant collaborations.

This work fulfils MoSCoW priorities (Table 1), enabling Music360 to support multi-party data sharing and collaborations without compromising confidentiality. Future efforts will expand SMPC to artists earning distribution and analysis (Scenario 2) and other possible cases.

Appendix A

Index	Description
R1	Data custody **OUGHT** to follow the status quo where feasible. Data providers (e.g., CMOs), thus, ought to retain management of the data they currently have custody and control over as to prevent data duplication in domains which are, as of yet, not production tested.
R2	Data providers **OUGHT** to make data in their custody, but owned by other parties, available to the ecosystem, within the reasonable scope of the project's data accessibility requirements.
R3	Data providers **OUGHT** to be empowered to choose which applications (e.g., the dashboard) they serve data to and receive data requests from via a whitelist. N.B., consensus among data providers upon a general whitelist would create the natural boundaries of the Music360 ecosystem but limits its scope until such a whitelist becomes expanded. Such consensus is not necessary but does limit attack vectors.
R4	Data providers **MUST**, where reasonably possible, integrate the software packages provided henceforth with their own systems to make a scope of data in their custody available to the ecosystem where such scope is reasonably required to meet the goals of the research project.
R5	CMOs **MUST** provide means for the following user groups to manage their data: creatives (e.g., artists), venues and policy makers.
R6	Data made available to the ecosystem **MUST** be done so in a secure manner in non-prototype systems (i.e., systems not using mocked data).
R7	Data owner(s) **MUST** be empowered to grant data access to requesting third parties within the ecosystem.
R8	Data owner(s) **MUST** be empowered to revoke data access to third parties.
R9	Data owner(s) **MUST** be warned about the risks of sharing data with third parties in the Music360 ecosystem **PRIOR TO** granting access to such parties.
R10	Data owner(s) **OUGHT** to give their consent and take full liability of the risks associated with data theft from third parties in the Music360 ecosystem.
R11	Access or revocation of data to and from the various parties of the ecosystem **MUST** be done in a timely and consistent manner.
R12	Communication between services in non-prototype systems **MUST** be done over TLS 1.2 or higher.
R13	Provisions against common web attacks (e.g., XSS, CSRF, Injections such as (but not necessarily limited to) SQL Injections, DoS/DDoS) **MUST** be taken by the various components of the ecosystem and documented at a later date in a comprehensive security policy.
R14	Ecosystem database(s), notably, but not limited to, described in D2.1 **OUGHT** to be partitioned in accordance with **R1**.
R15	Ecosystem database(s), notably, but not limited to, described in D2.1 **OUGHT** to be backed up and replicated at a reasonable frequency by the various data providing parties of the ecosystem.
R16	Authorization policies of the ecosystem database(s), notably, but not limited to, described in D2.1 **MUST** be as fine-grained in scope as reasonably possible given the current technological landscape and resource/computation budget.
R17	Ecosystem database(s), notably, but not limited to, described in D2.1 **MUST** be encrypted using cryptographically secure symmetric or asymmetric algorithms with sufficient collision entropy for the lifespan of the ecosystem; e.g., in the case that at year "n" RSA2048 is not predicted to provide sufficient collision resistance, RSA3072 must be phased in at year "n - (some reasonable time period)".
R18	Ecosystem database(s) **OUGHT** to phase in homomorphic encryption at such a time

Index	Description
	when it becomes strategically feasible to increase the security guarantees that the Music360 ecosystem can make.
R19	Encrypted ecosystem database(s) **OUGHT** to rotate keys at certain intervals under a strict policy in order to prevent data leaks and generally harden security.
R20	A penetration test **OUGHT** to be taken out by a sufficiently capable actor on a production system with mocked data such that any weaknesses or flaws may be identified and fixed proactively.
R21	Authorization and authentication of the ecosystem **MUST** be stateless and accomplished in accordance with the JWT standard (*RFC 7519, RFC 8725*).
R22	For all data providers in the ecosystem, JWTs generated **MUST** be signed with an asymmetric private key.
R23	For all data providers in the ecosystem, JWTs generated and used by requesting parties **MUST** provide details of the scope of data access, or can be used to derive scope of data access.
R24	All non-prototype services in the ecosystem **MUST** be highly available.
R25	All services implementing or working with standards **MUST** make a best effort to implement best common practices (BCP) where available, moreover, such services **MUST** also be compliant with data privacy and protection legislation within the jurisdictions they are available in (e.g., GDPR).
R26	Data providers **MUST** reach consensus on a policy for user lifecycle management e.g., Annex A ISO/IEC 27001:2013 section 9.2.
R27	Data providers **MUST** make data accessible in a manner compliant with the FAIR principles.

Appendix B

In DBEs, security requirements vary widely in complexity and urgency. To support this distinction, we examine typical requirement domains within DBEs, which include both traditional security concerns and DBE-specific demands. On the one hand, DBEs inherit general IT security challenges, such as access control, data integrity, authentication, data availablity and encryption at rest or in transit, that are well-understood and can be addressed using standard security mechanism. On the other hand, DBEs also introduce new challenges that emerge from decentralized governance, trust challenges, cross-border regulation, dynamic collaboration and possible competitive relationships. These more complex scenarios often stretch beyond the boundaries of what traditional mechanisms can offer, thereby motivating the use of advanced PPC technologies. Hence, our taxonomy thus serves not as an exhaustive list of all security concerns, but as a perspective for identifying when DBE-specific demand cross the threshold into the domain of PPC technologies. Below are six representative security patterns **[Sec-pt]** and their related requirement types **[RQTs]** observed in our case and validated through prior DBE literature (Lenkenhoff, K., 2018; Senyo, P. K., 2019; Tornjanski, V., 2021; Lindell, Y., 2020}.

1. **[Sec-pt 1] Secure multi-party data collaboration**

   - [RQT1] Confidential multi-stakeholder computation: Stakeholders can engage in joint computations without disclosing raw input values to each other, preserving the confidentiality of sensitive data.
   - [RQT2] Trust-limited collaboration: The computation framework must avoid reliance on centralized or trusted third parties, ensuring decentralized execution without exposing data.
   - [RQT3] Respect for distributed data ownership: The security design must ensure that data can only be included in computations when explicitly authorized by each independent stakeholder, reinforcing legal compliance.

2. **[Sec-pt 2] Privacy-preserving aggregation and result publication**

   - [RQT4] Protection against aggregation leakage: When publishing aggregate statistics, the security deisgn must ensure that individual contributions cannot be inferred, even with auxiliary external knowledge.
   - [RQT5] Controlled exposure of computation results: Published results must be processed to prevent reverse-engineering of private inputs, while still retaining analytical utility.

3. **[Sec-pt 3] Encrypted data processing**

   - [RQT6] Computation on processed inputs: The security design must support secure computation over processed (e.g., secret-shared, encrypted) data during processing, ensuring data protected end-to-end.

- [RQT7] Persistent encryption across lifecycle: Sensitive data must remain encrypted during storage, transit, and active use, avoiding plaintext exposure at any point in the pipeline.

4. **[Sec-pt 4] Collusion-resistant computation**

- [RQT8] Collusion-resistant protocol design: The security design must prevent coalitions of dishonest participants from inferring private data or manipulating computation outcomes.
- [RQT9] Resilience to insider threats: Computation must remain secure even when some participants are compromised or act maliciously, without degrading global guarantees.

5. **[Sec-pt 5] Verifiability without disclosure**

- [RQT10] Input/result verifiability without data disclosure: Security design should provide verification for the precise of computation results without gaining access to intermediate states or private inputs.

6. **[Sec-pt 6] Dynamic and resource-constrained security**

- [RQT11] Resource-constrained environments: The security design must remain functional in environments with limited computational capacity. This motivates the choice of cryptographic protocols that minimize resource consumption while still offering acceptable performance and security guarantees. In such settings, we aim for a practical trade-off between computational efficiency and implementation overhead, rather than adopting the most performant or most secure protocols in isolation.
- [RQT12] Compliance with evolving policies: The security design must adapt to varying or evolving privacy regulations across areas.

References

[1] Wang, Z., Cheung, S. C. S., & Luo, Y. (2016). Information-theoretic secure multi-party computation with collusion deterrence. *IEEE Transactions on Information Forensics and Security*, *12*(4), 980-995.

[2] Zhao, C., Zhao, S., Zhao, M., Chen, Z., Gao, C. Z., Li, H., & Tan, Y. A. (2019). Secure multi-party computation: theory, practice and applications. *Information Sciences*, *476*, 357-372.

[3] "IEEE Recommended Practice for Secure Multi-Party Computation," in IEEE Std 2842-2021 , vol., no., pp.1-30, 5 Nov. 2021, doi: 10.1109/IEEESTD.2021.9604029. keywords: {IEEE Standards;Security;Computation theory;IEEE 2842;MPC;secure multi-party computation;technical framework},

[4] Bian, S., Jiang, W., & Sato, T. (2021, December). Privacy-preserving medical image segmentation via hybrid trusted execution environment. In *2021 58th ACM/IEEE Design Automation Conference (DAC)* (pp. 1347-1350). IEEE.

[5] Zhang, Q., Xin, C., & Wu, H. (2021). Privacy-preserving deep learning based on multiparty secure computation: A survey. *IEEE Internet of Things Journal*, *8*(13), 10412-10429.

[6] HAN Wei-Li, SONG Lu-shan, R.W.q.L.G.p.W.Z.x. (2023): Secure multi-party learning: From secure computation to secure learning. *CHINESE JOURNAL OF COMPUT ERS 46(7),* 1494–1512

[7] Guo, J., Wang, Q., Xu, X., Wang, T., & Lin, J. (2021). Secure multiparty computation and application in machine learning. *Journal of Computer Research and Development*, *58*, 2163-2186.

[8] Addanki, S., Garbe, K., Jaffe, E., Ostrovsky, R., & Polychroniadou, A. (2022, September). Prio+: Privacy preserving aggregate statistics via boolean shares. In *International Conference on Security and Cryptography for Networks* (pp. 516-539). Cham: Springer International Publishing.

[9] Keller, M., Pastro, V., & Rotaru, D. (2018, March). Overdrive: Making SPDZ great again. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 158-189). Cham: Springer International Publishing.

[10] Amini, R., Fesq, L., Mackey, R., Mirza, F., Rasmussen, R., Troesch, M., & Kolcio, K. (2021, March). FRESCO: A framework for spacecraft systems autonomy. In *2021 IEEE Aerospace Conference (50100)* (pp. 1-18). IEEE.

[11] Keller, M., Orsini, E., & Scholl, P. (2016, October). MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 830-842).

[12] Rathee, M., Zhang, Y., Corrigan-Gibbs, H., & Popa, R. A. (2024, May). Private analytics via streaming, sketching, and silently verifiable proofs. In *2024 IEEE Symposium on Security and Privacy (SP)* (pp. 3072-3090). IEEE.

[13] Wieringa, R., & Moralı, A. (2012, May). Technical action research as a validation method in information systems design science. In *International Conference on Design Science Research in Information Systems* (pp. 220-238). Berlin, Heidelberg: Springer Berlin Heidelberg.

[14] Lenkenhoff, K., Wilkens, U., Zheng, M., Süße, T., Kuhlenkötter, B., & Ming, X. (2018). Key challenges of digital business ecosystem development and how to cope with them. *Procedia Cirp*, *73*, 167-172.

[15] Senyo, P. K., Liu, K., & Effah, J. (2019). Digital business ecosystem: Literature review and a framework for future research. *International journal of information management, 47*, 52-64.

[16] Tornjanski, V., Knežević, S., Ljubanić, D., Glišić, V., Žižić, D., & Travica, J. (2021, September). Towards secured digital business ecosystems: From threats to opportunities. In *E-business technologies conference proceedings* (Vol. 1, No. 1, pp. 1-14).

[17] Lindell, Y. (2020). Secure multiparty computation. *Communications of the ACM*, *64*(1), 86-96.

[18] Freedman, M. J., Nissim, K., & Pinkas, B. (2004). Efficient private matching and set intersection. *EUROCRYPT*.

[19] De Cristofaro, E., & Tsudik, G. (2010). Practical private set intersection protocols with linear complexity. *Financial Cryptography*.

[20] Dong, C., Chen, L., & Wen, Z. (2013, November). When private set intersection meets big data: an efficient and scalable protocol. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 789-800).