Master Data for MAaaS:

The sources collectively present information on two distinct, highly technical domains: **Secure Multi-Party Computation (MPC)** and **AI Agent Development leveraging the Model Context Protocol (MCP)**. The first source provides a foundational and comprehensive overview of MPC, detailing its evolution, core protocols like **Yao's Garbled Circuits** and **GMW**, methods for improving **efficiency** (e.g., Oblivious RAM), and techniques to achieve robustness against **malicious adversaries** (e.g., Cut-and-Choose). The remaining sources focus on modern AI workflows, introducing **AI Agents** and the **Model Context Protocol (MCP)**, which facilitates the agents' ability to use **external tools and resources**. Specifically, these texts describe agent design patterns like **Reflection/Refinement** and **Context Augmentation**, detail the client-server architecture of MCP for **tool integration** in the **Agent Development Kit (ADK)**, and include an **audit report** for an MPC library, highlighting security flaws such as **timing side-channel vulnerabilities** in cryptographic functions.

This report synthesizes the provided sources as Master Data for Agent Creation, covering the foundations of agent architectures, security frameworks, underlying protocols, tooling, and critical challenges in deploying secure, complex agent systems.

Master Data Report for Agent Creation

I. Core Agent Foundations and Architectures

A. Agent Definition and Role AI Agents are systems that utilize a Large Language Model (LLM) to interpret natural language instructions, interact with their environment, and take actions toward a goal. They represent an abstraction shift where users specify desired outcomes, not actions, and the system autonomously determines the execution path. The user's role evolves from operator to orchestrator or supervisor.

B. Single-Agent Architectural Patterns Single-agent architectures rely on one agent to manage the entire workflow, encompassing reasoning, planning, execution, and tool interaction.

Pattern	Primary Function / Use Case	Key Features
Tool-using Agent	Simple automation, prototypes, handling tasks requiring external APIs (e.g., customer support using a billing API),.	Logic stays in the agent; heavy lifting is delegated to tools, often abstracted via a Model Context Protocol (MCP) layer. Recommended starting point for MVPs,.

Memory-augmented Agent	Tasks requiring awareness of past context, such as user interactions, historical data, or external states (e.g., personalized reminders).	Queries past context from a vector memory store.
Planning-agent	Tasks that require sequential, coordinated actions and cannot be completed in a single step (e.g., multi-step SaaS onboarding),,.	Generates a multi-step plan, tracks execution, and adapts as dependencies require.
Reflection-agent	Systems requiring continuous improvement and self-correction (e.g., adjusting a trading strategy based on past performance).	Stores results, compares them to goals/metrics, and updates its strategy via a feedback loop,,.

C. Multi-Agent Architectural Patterns Multi-agent architectures involve multiple specialized agents collaborating to complete complex workflows, enabling scalability, parallelism, and precision.

Pattern	**Primary Function / Use Case**	**Key Coordination,**
Supervisor	Complex tasks requiring delegation to specialized workers (e.g., hospital appointment scheduling),,.	A single lead agent breaks tasks into sub-tasks and delegates them, managing order and context flow.
Hierarchical	Tasks too complex or broad for a single supervisor, requiring specialization across teams or domains (e.g., enterprise document processing pipeline),.	Layers of coordination: top-level delegates to mid-level, which delegates to lower-level workers.
Competitive	Tasks where diverse solutions or redundancy are beneficial (e.g., generating marketing copy),.	Multiple agents work independently on the same problem; a separate evaluator selects the best submission based on predefined criteria.

II. MPC and Cryptographic Security Frameworks

A. Secure Multi-Party Computation (MPC) MPC enables mutually distrusting parties to jointly compute a function on their secret inputs without revealing the inputs themselves,. MPC evolved from a theoretical concept in the 1980s to a practical tool today.

B. Core Security Models (MPC) Security is defined using the **real-ideal paradigm**, comparing a real-world protocol execution to an ideal world where a completely trusted third party (functionality T) handles the computation,.

Model	**Adversary Type**	**Security Guarantee**	**Note**
Semi-Honest	Passive; follows protocol but tries to learn inputs from received messages.	Provides privacy and correctness under the assumption all parties execute the protocol honestly,.	Often serves as the basis for more robust protocols.
Malicious	Active; can deviate arbitrarily from the protocol to violate privacy or correctness.	Provides the strongest cryptographic guarantees.	Historically incurred significant performance penalties.
Security with Abort	Weaker malicious security.	A malicious party who learns the output can prevent honest parties from receiving it,.	Output fairness is not guaranteed.
Covert Security	Active; cheating is deterred by a fixed probability ϵ of being caught,.	More efficient than malicious security.	
PVC	Active; similar to covert.	If cheating is detected, the honest party can produce a publicly verifiable proof ("certificate of cheating"),.	Reduces deterrence factor risk by introducing public accountability.

C. Preserving User Agency in Crypto Systems In agent-driven crypto systems (offchain synthesis, onchain execution), user agency requires rethinking trust, delegation, and privacy,.

Framework / Standard	**Purpose**	**Method**
Agent Permissioning	Scoped delegation and fine-grained control over agent actions.	Uses account abstraction, MPC-based key control, session-limited delegation, and policy-based permissions (e.g., MetaMask Delegation

		Toolkit, Coinbase AgentKit/x402, Biconomy DAN).
Intent-centric Infrastructure	Shifts the focus from defining *actions* to defining desired *outcomes*,.	Decentralized network of solvers competes to fulfill user-defined intents efficiently (e.g., NEAR Intents, Anoma).
Verifiable Execution Layers	Provides cryptographic and incentive-aligned mechanisms for validating agent behavior.	Staked validators attest to offchain execution (e.g., EigenCloud AVS, AVA Protocol); **Zero-knowledge (ZK) proofs** verify computation correctness without revealing inputs,.

III. MPC Protocols and Primitives

A. Fundamental MPC Protocols

Protocol	Protocol Type	Security Model	Description
Yao's GC	2PC	Semi-Honest	Uses Boolean circuits; achieves computation in constant rounds,.
GMW	Multi-party	Semi-Honest	Uses additive shares; rounds scale with circuit depth,.
BGW	Multi-party	Honest Majority	Uses Shamir secret sharing over an arithmetic circuit; $2t < n$ parties honest,.
Authenticated Sharing	Multi-party	Malicious	Splits execution into **preprocessing** (generating Beaver multiplication triples) and **online** phases. Implemented via BDOZ or SPDZ,.
Authenticated Garbling	2PC/Multi-party	Malicious	Combines authenticated secret-sharing with BMR/GC to achieve high performance (e.g., 0.8 million malicious AND gates/sec),.

B. Essential Cryptographic Primitives

Primitive	Functionality	Relevance to Agents

Oblivious Transfer (OT)	R gets x_b from S without S learning b or R learning x_{1-b}.	Theoretically equivalent to MPC; fundamental building block for GC protocols.
Zero-Knowledge Proofs (ZK)	Prover shows $C(x)=1$ without revealing x.	Used to enforce correct protocol execution in the GMW compiler,, and for verifiable reasoning in privacy-preserving LLM services,.
Secret Sharing	Splits secret s into n shares such that $t-1$ shares reveal nothing, but t shares allow reconstruction.	Core mechanism underlying many MPC approaches (e.g., GMW, BGW, BDOZ/SPDZ),,.

C. Cryptographic Implementation Risks (Coinbase cb-mpc Audit) High-stakes MPC libraries require rigorous auditing. Vulnerabilities identified in the Coinbase cb-mpc library include:

Finding	Severity	Description
Small Subgroup Attacks (CBS-02-004)	**High**	ECC-Refresh-MP fails to validate that received elliptic curve points belong to the prime-order subgroup, potentially compromising security via key compromise or Denial of Service,,.
Paillier Key Generation (CBS-02-006)	Medium	Lacks essential algebraic checks required for the cryptosystem's security proofs,,.
Missing Parameter Checks (CBS-02-003)	Low	ECC-Refresh-MP lacks checks to ensure all parties agree on session identifiers and keys, violating protocol security assumptions in open environments,,.
Non-Deterministic Nonce (CBS-02-001)	Info	Ed25519 signing deviates from standard specification by using random nonces, risking nonce reuse and compatibility issues,,.
Variable-time Branching (CBS-02-002)	Info	Modular inversion function uses variable-time behavior, potentially exposing the implementation to timing side-channel attacks,,.

IV. Agentic Tooling and Interoperability

A. Model Context Protocol (MCP) MCP is an open standard that acts as a universal connection mechanism, simplifying how LLMs gain context, execute actions, and interact with systems.

Component	**Function**	**Role in Agent Workflows**
MCP Server	Exposes actionable **tools** (functions), **interactive templates** (prompts), and **data** (resources),.	Provides parameterized prompts for *prompt chaining*, supplying the LLM with the next instruction or dynamic tools,.
MCP Client	An LLM host application or AI Agent (e.g., ADK Agent) that consumes resources from the server.	Orchestrates the agentic workflow: manages the reasoning loop, coordinates tool execution, and tracks session state.
Nesting/Composition	MCP servers act as clients to other MCP servers.	Enables modularity, delegation, and complex orchestration (like microservices for agents),.

B. Agent Development Tools and Patterns

Tool/Pattern	**Description**	**Related Frameworks**
Agent Development Kit (ADK)	A framework/library (e.g., Python, Java) for building agents. The McpToolset class bridges ADK agents to external MCP servers.	Google ADK, often interacting with servers via npx (for local Stdio) or Streamable HTTP (for remote/scalable deployment),.
Agent-to-Agent (A2A) Protocol	A standardized communication layer for agent discovery, task lifecycle, and secure messaging.	Google A2A Protocol, MIT NANDA, ERC-8004,.
Evaluator-Optimizer	Iteratively refines LLM responses by using an *Evaluator* to assess output against requirements and an *Optimizer* (Generator) to implement concrete improvements,.	Useful for generating critical software code or complex analytical work.
Context-Augmentation	Dynamically expands the LLM's context beyond pre-trained knowledge by calling external tools and systems,.	Often standardized by MCP.

Prompt-Chaining	Decomposes complex tasks into sequential subtasks, where the output of one specialized prompt becomes the input for the next,.	Enhances transparency, controllability, and flexibility.

V. Implementation and Optimization Techniques (MPC)

Area	**Technique**	**Impact/Efficiency Gains**
Garbling Cost	**Half Gates**,	Reduces AND gates to two ciphertexts, compatible with FreeXOR, achieving bandwidth optimality for its class,.
Computational Cost	**FreeXOR**	Computes XOR gates without ciphertexts or garbled tables by enforcing a global key offset Δ,.
Bandwidth	**Garbled Row Reduction (GRR3)**	Reduces transmitted ciphertexts per gate (e.g., from four to three),.
Circuit Design	**Low-Depth Circuits**	Critical for protocols like GMW, where execution time depends heavily on circuit depth (number of AND gates in sequence),.
Data Structures	**Oblivious RAM (ORAM)**	Enables sublinear memory access costs for private indexes $a[]$ where simple linear scan is prohibitive,. Designs like Floram (Function-secret-sharing Linear ORAM) offer substantial concrete performance improvements,.
Overall Performance	**Modern MPC Frameworks**	Have achieved performance gains of 3–9 orders of magnitude over the past decade. Malicious secure 2PC is now possible at over 0.8 million gates per second,.

VI. Security Challenges in Cross-Domain Multi-Agent Systems

The security agenda for systems where autonomous agents cooperate across organizational trust boundaries is defined by novel behavioral and data-centric challenges,.

Challenge ID	**Name**	**Problem Description**	**Recommended Countermeasure Direction**

C1	**Unvetted Dynamic Grouping**	New, unvetted models or agents join at runtime, blurring trust boundaries and leading to unpredictable, unvetted group activities.	Trust-adaptive Dynamic Teaming: Agents maintain a differentiable trust ledger for peers, quarantining low-trust peers.
C2	**Collusion Control**	Agents from separate domains coordinate hidden agendas, often exploiting split-knowledge cartels or covert channels (e.g., steganography) to evade audit,.	Adversarial Multi-agent Training for Collusion Resistance: Training systems against synthetic collusion scenarios until collusion yields no net gain.
C3	**Conflicting Incentives & Goals**	Agents prioritize their owner's local interests over the collective goal, amplified by the absence of a shared trust anchor in cross-domain hierarchies.	Hierarchical Conflict Arbitration via Meta-LLM Controller: A meta-LLM monitors, arbitrates, and generates resolving instructions for conflicts between agents.
C4	**Distributed Self-tuning Misalignment**	Locally applied, unorchestrated reward signals propagate across distributed fine-tuning loops, causing gradual misalignment from the safe or intended global objective.	Cross-domain Reward Alignment via Adaptive Credit Assignment: Aligning local rewards with the global objective using a shared critic model that computes tailored reward signals.
C5	**Cross-domain Provenance Obscurity**	The entanglement of data inputs within LLM latent features and separated domain logs prevent the unified tracing of malicious influence, hindering attribution,.	Neural Provenance Tracking with Embedded Signatures: Agents embed subtle signatures (watermarks) in generated content, allowing a decoder model to reconstruct the timeline of contributions.
C6	**Cross-domain Context Bypass**	Fragmented context across multiple agents allows sensitive data to be reconstructed or policies to be violated through	Session-level Semantic Firewalls: A dedicated firewall LLM ingests the entire multi-agent dialogue in a sliding window to detect

		incremental, seemingly benign queries,.	when composite context breaches policy.
C7	**Inter-domain Confidentiality & Integrity**	High confidentiality (using FHE or MPC) can create an integrity gap, allowing a malicious user to forge the output of a "blind" service, as no party can cryptographically attest to the original plaintext result,.	Verifiable Reasoning with Privacy: The agent emits an encrypted answer and a public proof sketch (e.g., lightweight ZK proofs or authenticated traces) to confirm semantic correctness without revealing inputs.